

NASA-CR-197396

Purchase Order F435025
MCR-92-5014

100
42046
204p

30 October 1992

Final Report -
SHM Design
Methodology

Rocket Engine Condition Monitoring System (RECMS)

(NASA-CR-197396) ROCKET ENGINE
CONDITION MONITORING SYSTEM
(REDMS): SHM DESIGN METHODOLOGY
Final Report, 18 Nov. 1991 - 31
Oct. 1992 (Martin Marietta Corp.)
204 p

N95-70761

Unclass

Z9/20 0042046

Author: Glen Campbell, Stephen Johnson, Maxine
Obleski, Ron Puening

Prepared for: Pratt & Whitney
West Palm Beach, FL 33410-9600

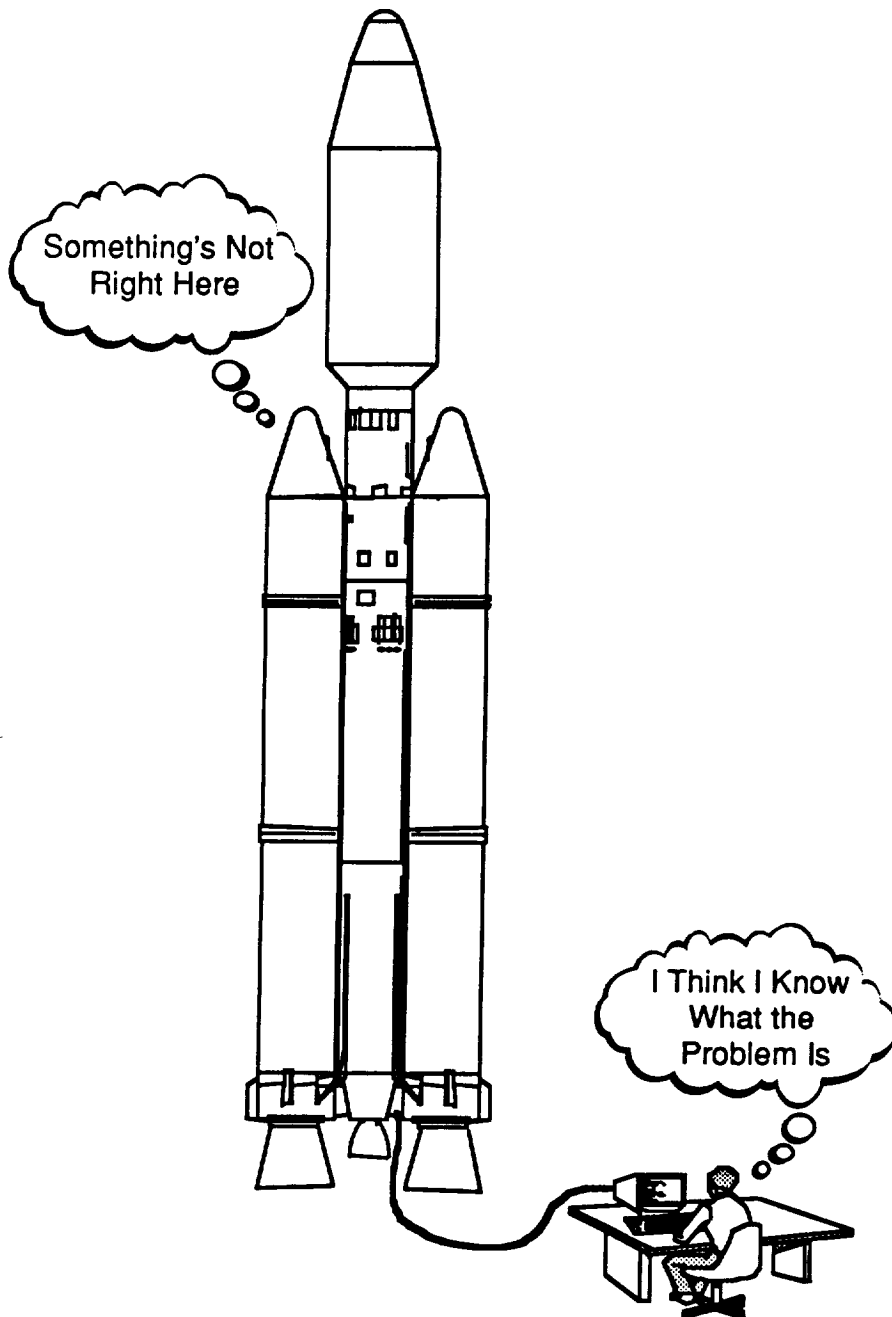
Period of Performance: 11/18/91 thru 10/31/92

MARTIN MARIETTA

Space Launch Systems Company
P. O. Box 179
Denver, CO 80201

SHM Design Methodology

(Draft Rev Sept 25 1992)



MARTIN MARIETTA

Space Launch Systems Company
PO Box 179
Denver, CO 80201

1

2

3

4

5

6

Short Introduction and Scope

The objective of the Systems Health Management Methodology is to delineate the techniques, methods, and reasoning behind design of a system health management system for a launch vehicle. Systems health management consists of all the design provisions to both prevent the occurrence of faults and mitigate the effect of faults that do occur.

Due to funding cuts, the document scope was reduced to detailed text for the introduction, initial requirements, conceptual design, and preliminary design phases of SHM design. The detail design phase and subsequent sections are outlined, and can be expanded into full textual format in follow on contract activity if funding materializes.

)

.

.

)

.

.

)

Preface

This document was prepared under Purchase Order #F435025 of contract F04611-89-C-0020 administered by:

United Technologies Corporation
Pratt & Whitney
Government Engines & Space Propulsion
PO Box 109600
West Palm Beach, FL 33410-9600

—

.

.

—

.

.

—

Table of Contents

Preface.....	ii
Table of Contents	iii
List of Tables and Figures.....	viii
1.0. System Health Management Methodology Introduction	1
1.1. Statement of Problem	1
1.2. Scope	3
1.2.1. Initial Definitions	3
1.2.2. Scope of System Health Management	3
1.2.3. Scope of the SHM Methodology Document.....	6
1.3. Objective	7
1.4. Structure of this Document	7
2.0. System Health Management Issues and Approach	8
2.1. SHM Design Approach	8
2.1.1. Infinity of the Fault Domain.....	8
2.1.2. Quantitative Criteria	12
2.1.3. Qualitative Criteria.....	15
2.1.4. Summary	18
2.2. System Process Assumptions	19
2.2.1. Role of SHM in Systems Engineering and Integration	19
2.2.2. Integrated Product Teams/Concurrent Engineering	24
2.2.3. Spiral and Waterfall Models	25
2.2.4. Rapid Prototyping	26
2.3. Overview of the SHM Design Methodology	27
2.3.1. Quantitative Requirements.....	30
2.3.2. Qualitative Requirements.....	31
2.3.3. Fault Set Definition	31
2.3.4. Fault Analysis and Modeling	32
2.3.5. System Design.....	34
2.3.6. Verification and Validation	34
2.3.7. Time Phasing of the SHM Design Process	35
2.4. System Specific Implications	38
2.4.1. Clean Sheet Versus Retrofit Design.....	38
2.4.2. One of a Kind Versus Production Quantities	39
3.0. SHM Design Process.....	41
3.1. System Health Management Initial Requirements Phase	41
3.1.1. SHM Initial Requirements Phase Objective.....	42
3.1.2. SHM Initial Requirements Phase Major Activities.....	42
3.1.3. SHM Initial Requirements Development Approach	44
3.1.3.1. System Requirements = Top Level Customer Demands	44
3.1.3.1.1. Integrated Product Development.....	45
3.1.3.1.2. Quality Function Deployment.....	46
3.1.3.1.3. Goals, Products, Constraints	48
3.1.3.1.4. Requirements Generation and Prioritization	49

—

.

.

—

.

.

—

3.1.3.2.	System Concept Development	50
3.1.3.2.1.	Figures of Merit Development	50
3.1.3.2.2.	Definition of Trade Studies	53
3.1.3.2.3.	Concept Evaluation	54
3.1.3.2.4.	Integrating SHM With Operations and Maintenance Concept.....	56
3.1.3.3.	SHM Initial Requirements Development.....	57
3.1.3.3.1.	Requirements Versus Design Guidelines.....	58
3.1.3.3.2.	SHM Requirements	58
3.1.3.3.3.	Development of Fault Tolerance Requirements.....	62
3.1.3.3.4.	Hardware Requirements.....	66
3.1.3.3.5.	Software Requirements	67
3.1.3.3.6.	Operational Requirements.....	68
3.1.3.3.7.	System Reliability and Maintainability Apportionment	68
3.1.4.	SHM Initial Design Requirements Phase Summary	70
3.2.	SHM Conceptual Design Phase	71
3.2.1.	Conceptual Design Phase Objective	72
3.2.2.	Conceptual Design Phase Major Activities.....	73
3.2.2.2.	Major Analyses	77
3.2.2.2.1.	Time to Criticality Analysis	77
3.2.2.2.2.	Fault Set Refinement Analysis.....	77
3.2.2.2.3.	Fault Injection Methods Analysis	77
3.2.3.	Conceptual Design Development Approach	78
3.2.3.1.	System Level Design Inputs to HMS Design.....	78
3.2.3.1.1.	Subsystem Concepts Without Health Management As A Primary Focus.....	78
3.2.3.1.2.	Subsystem Design and Figures of Merit	79
3.2.3.2.	Conceptual Design Requirements	80
3.2.3.2.1.	Subsystem Time to Criticality Requirements.....	82
3.2.3.2.2.	GIMADS Program Contribution To SHM Conceptual Design Requirements	82
3.2.3.2.3.	Dependable System Attribute Parameter Formulation and Allocation	84
3.2.3.2.4.	Performance, Safety, and Other Requirements for SHM Conceptual Design.....	85
3.2.3.2.5.	SHM Software Requirements Development.....	86
3.2.3.2.6.	SHM Operational Requirements	89
3.2.3.2.	Analyses & Design.....	90
3.2.3.2.1.	Time to Criticality Analysis	90
3.2.3.2.2.	Conceptual Design Preliminary Fault Accommodation List.....	95
3.2.3.2.3.	Functional Fault Matrix.....	97
3.2.3.2.4.	ECR/FCR Architecture	100
3.2.3.2.5.	Life Usage and Fault Prediction.....	103
3.2.3.2.6.	Verification and Validation Plan.....	103

—

.

.

—

.

.

—

3.2.3.2.7.	HMS Conceptual Design.....	109
3.2.3.3.	Preliminary Design HMS Requirements.....	109
3.2.3.3.1.	Test Bed and Real Time Simulation	109
3.2.3.3.2.	H/W, S/W, Operations Examples of Requirements Types	111
3.2.4.	SHM Conceptual Design Phase Summary.....	112
3.3.	Preliminary Design Phase	113
3.3.1.	Preliminary Design Phase Objective.....	115
3.3.2.	Preliminary Design Phase Major Activities.....	115
3.3.3.	Preliminary Design Approach.....	118
3.3.3.1.	Inputs to Preliminary Design Process	118
3.3.3.2.	Trade Studies.....	118
3.3.3.2.1.	General Parameter Selection Criteria.....	118
3.3.3.2.2.	Non-Economic Justifications for Parameters.....	119
3.3.3.2.3.	Parameter Characteristics.....	119
3.3.3.2.4.	Fault Accommodation Cost Effectiveness Trades	121
3.3.3.2.4.1.	Flight Fault Tolerance Parameter Selection (Flight Failure Prevention Cost Effectiveness Analysis).....	122
3.3.3.2.4.2.	Cost Effectiveness Based on Factors Other Than Flight Loss Prevention	124
3.3.3.2.4.3.	Complexities.....	125
3.3.3.2.5.	Sensor Selection	127
3.3.3.3.	Preliminary Design Analyses.....	128
3.3.3.3.1.	FMEA and the Gathered Fault Combination Method (GCFM).....	128
3.3.3.3.2.	Fault and Fault Propagation Modeling.....	130
3.3.3.3.3.	False Alarm Analyses.....	136
3.3.3.3.4.	Time to Criticality - Preliminary Design Analyses.....	139
3.3.3.3.5.	Cost Effectiveness and Reliability Analysis	141
3.3.3.4.	Preliminary SHM Design.....	146
3.3.3.4.1.	Development of the Fault Accommodation List.....	146
3.3.3.4.2.	ECR/FCR Refinement.....	149
3.3.3.4.3.	Fault Prediction, Detection, Isolation, Response Implementation.....	153
3.3.3.4.4.	SHM Implementation Issues	159
3.3.3.5.	Simulation/Test Bed Design.....	160
3.3.3.6.	Detail Design Requirements.....	160
3.3.4.	Preliminary Design Phase Summary.....	160
3.4.	SHM Detail Design Phase.....	163
3.4.1.	Detail Design Phase Objective.....	163
3.4.2.	Detail Design Phase Major Activities	163
3.4.3.	Detail Design Approach	163
3.4.3.1.	HMS Model Refinement.....	163
3.4.3.2.	Quantitative Threshold Determination.....	163
3.4.3.3.	Formal Design.....	163

—

.

.

—

.

.

—

3.4.3.4.	Fault Injection Into Detail Design	163
3.4.3.5.	Detail Design Practice for Fault Avoidance.....	163
3.4.3.6.	Final FMEA.....	163
3.4.3.7.	HMS/System Integration Detail Design Issues	163
3.4.3.8	Detailed Data Management Plan.....	163
3.4.4	Subsystem Detail Design Issues.....	163
3.4.5.	HMS Design Support Planning (Training, Personnel, Etc)	163
3.4.6.	Requirements for Fabrication and Test Phase (Test Plan)	163
3.4.7.	Detail Design Phase Summary	163
3.5.	SHM Fabrication and Test Phase	163
3.5.1.	Fabrication & Test Phase Objectives	163
3.5.2.	Fabrication & Test Phase Major Activities	163
3.5.3.	Fabrication & Test Phase Approach.....	163
3.5.3.1.	Fabrication of HW.....	163
3.5.3.2.	Verification and Validation	163
3.5.3.2.1.	Analysis.....	163
3.5.3.2.2.	Testing.....	163
3.5.3.2.2.1.	Component and Box Level.....	163
3.5.3.2.2.2.	Subsystem Level	164
3.5.3.2.2.3.	System Level Testing	164
3.5.3.2.3.	Formal Proof	164
3.5.3.2.4.	Simulation	164
3.5.3.3.	Threshold Adjustment.....	164
3.5.3.4	Preliminary Operations Planning	164
3.5.4.	Fabrication and Test Phase Summary	164
3.6.	System Deployment and Design Feedback Phase.....	164
4.0.	Recommendations	164
4.1.	Design Process Tools	164
4.2.	Design Organizational Issues	164
4.3.	Technology Development	164
4.4	Process Development	164
Appendix A.	References	165
Appendix B.	List of Acronyms	169
Appendix C.	Definitions	171
Appendix D.	Methodology Tools.....	179
Tools for Initial Requirements Phase		179
Requirements Development		179
Quality Function Deployment.....		180
Formal Requirements Development.....		180
Tools for the Conceptual Design Phase		181
Design Synthesis		181
Data Flow Methodology.....		182
Testability Analysis.....		183
FMEA/FMECA Tools.....		184
Cost vs Reliability Modeling.....		185
Reliability Modeling		185

—

.

.

—

.

.

—

Performance Modeling Tools.....	186
Design Verification and Validation.....	187
Formal Verification.....	187
Design Database for HMS.....	188
Appendix E. Methodology Summary.....	189

—

.

.

—

.

.

—

List of Tables and Figures

Figure 1.1-1	The Four Elements of the Health Management System.....	2
Figure 1.1-2.	HMS and DSE are Subsets of SHM.....	6
Figure 2.1-1	Fault Coverage	10
Figure 2.1-2.	Subsystem Sources of Failure	14
Figure 2.2-1.	SHM/System Design Overlap	19
Figure 2.2-2.	Systems Engineering Process.....	21
Figure 2.2-3.	Requirements/Implementation Iteration.....	23
Figure 2.2-4.	The Design Spiral.....	25
Figure 2.2-5.	The Waterfall View of the Design Process	26
Figure 2.3-1a.	Overview of SHM Task Part 1	28
Figure 2.3-1b.	Overview of SHM Task Part 2	29
Figure 3.1-1.	Initial Requirements Phase is Just the Beginning	41
Figure 3.1-2.	The System Health Management Initial Requirements Phase	43
Figure 3.1-3.	Joint Effort Requirements Development.....	45
Figure 3.1-4.	QFD Overview	47
Figure 3.1-5.	Constraints.....	48
Figure 3.1-6.	Figures of Merit.....	51
Figure 3.1-7.	A-1 Matrix Example.....	52
Figure 3.1-8.	A-3 Matrix.....	54
Figure 3.1-9.	Integrating HMS With Operations, Maintenance and Other Ground Systems.....	57
Figure 3.1-10.	Qualitative & Quantitative Requirements.....	58
Figure 3.1-11.	Dependable System Attribute Tree	60
Figure 3.1-12.	NLS Robustness Tree.....	60
Figure 3.1-13.	Requirements Phase Goals & Activities	70
Figure 3.2-1.	System Level Decomposition.....	71
Figure 3.2-2.	SHM Conceptual Design Flow	76
Figure 3.2-3.	Main Elements - NLS Top Level Functional Flow.....	79
Figure 3.2-4.	Requirements Relationship to Conceptual Design Phase Flow	81
Figure 3.2-5.	GIMADS Contributions to SHM Requirements Development.....	83
Figure 3.2-6.	Maintainability Branch of Dependable System Attribute Tree.....	85
Figure 3.2-7.	Software Requirements Methodology Classifications	88
Figure 3.2-8.	Magellan EPS - AACS Interaction.....	91
Figure 3.2-9.	A Top-Level Time to Criticality Matrix	92
Figure 3.2-10.	NLS/CTV Mission Event Sequence.....	93
Figure 3.2-11.	Functional Fault Matrix.....	98
Figure 3.2-12.	Example of Fault Containment Region Use.....	101
Figure 3.2-13.	Some Methods of Fault Containment.....	102
Figure 3.2-14.	The Cost of Verification and Validation	104
Figure 3.2-15.	Methods of Validation and Verification.....	105
Figure 3.2-16.	An Example V & V Plan.....	107
Figure 3.2-17.	Time Tagging HM Activities	110
Figure 3.2-18	Provisions for Testbed/Simulation Expansion	111

—

.

.

—

.

.

—

Figure 3.3-1.	SHM Preliminary Design Overview	114
Figure 3.3-2.	Mutual Systems Design and SHM Design Decisions	115
Figure 3.3-3.	Failure Criticality and Compensable Failure Coverage	122
Figure 3.3-4.	Typical Parameter Economic Effectiveness Assessment.....	123
Figure 3.3-5.	System Coupling Complicates Fault Isolation	126
Figure 3.3-6.	Failure Criticality Categorization Process.....	130
Figure 3.3-7.	Fault Propagation Time to Criticality.....	131
Figure 3.3-8.	Avionics Cooling Cause-Consequence Diagram	132
Figure 3.3-9.	Time to Criticality for Fault Propagation Process.....	133
Figure 3.3-10.	FTA #2 Main Oxidizer Valve Fails Closed	134
Figure 3.3-11(a).	Fault Propagation-Main Oxidizer Valve Fails Closed	135
Figure 3.3-11(b).	Fault Propagation-Valve Fails Closed.....	136
Figure 3.3-12.	False Alarm Analysis Execution	137
Figure 3.3-13.	SHM Design Impact of False Alarm Analysis.....	138
Figure 3.3-14.	Allocating Components of Time to Criticality	140
Figure 3.3-15.	Designer Breakdown for Time to Criticality.....	141
Figure 3.3-16.	The Fault Accommodation List.....	147
Figure 3.3-17.	Fault Taxonomy Tree	148
Figure 3.3-18.	Detections Are Mapped to Fault	155
Figure 3.3-19.	Fault Response Sequence	156
Figure 3.3-20.	A Typical Feedback Control Loop Set.....	158
Figure D-1.	Example of a Data Flow Diagram.....	183
Table 2.1-1.	Summary of Qualitative and Quantitative Criterion	19
Table 3.1-1	Health Management Data Handling Plan.....	56
Table 3.1-2.	Fault Classes.....	62
Table 3.1-3.	Fault Tolerance Requirement.....	65
Table 3.1-4.	Fault Cause Classes within Implementation	66
Table 3.1-5.	Some System Human Roles	68
Table 3.1-6.	Reliability and Maintainability Apportionment	69
Table 3.2-1.	Conceptual Design Objective.....	73
Table 3.2-2.	Conceptual Design Activities and Products	74
Table 3.2-3.	HM Benefits for Launch Vehicles	86
Table 3.2-4.	Software Requirements Categories	87
Table 3.2-5.	Preliminary Fault Accommodation List.....	96
Table 3.2-6.	Requirements Examples for Preliminary Design	112
Table 3.2-7.	Key activities for Conceptual Design Phase	112
Table 3.3-1.	SHM Preliminary Design Activities and Products.....	116
Table 3.3-2.	Cost Savings Contributions From Parameters By Use.....	124
Table 3.3-3.	Typical Sensor Requirements.....	128
Table 3.3-4.	Gathered Fault Combination Method (GFCM) Steps.....	129
Table 3.3-5.	System Fault Status Perception.....	136
Table 3.3-6	Major Factors for SHM Cost Benefit Analysis.....	143
Table 3.3-7.	Typical Digital Fault Handling Sequence	149
Table 3.3-8.	Hypothetical Five Layered ECR (from[SIEW 91])	150

—

.

.

—

.

.

—

Table 3.3-9.	Typical Techniques For Digital Processor Fault Confinement and Detection	151
Table 3.3-10.	Factors To Consider In ECR/FCR Design	152
Table 3.3-11.	Preliminary Design Phase Summary	161
Table 3.3-12.	Preliminary Design Review Questions	162
Table E-1.	Products and Activities of the Initial Requirements Phase	189
Table E-2.	Conceptual Design Products	190
Table E-3.	SHM Preliminary Design Activities and Products.....	191
Table E-4.	Products of Detailed Design Phase	192
Table E-5.	Products of Fabrication and Test Phase	192
Table E-6.	Products of Deployment and Operations Phase	193
Table E-7.	SHM Design Review Questions.....	193

—

.

.

—

.

.

—

1.0. System Health Management Methodology Introduction

1.1. Statement of Problem

As the complexity and autonomy of systems increases, it becomes increasingly difficult to build them in a dependable manner. In addition, the cost of failure has greatly increased as well, due to the increased cost of these vehicles and systems. These factors have led to an increasing awareness of the importance of systematically dealing with the problem of system failure. Within the space launch vehicle community, these concerns have crystallized under the term, "Vehicle Health Management" (VHM). As will be apparent, the term "System Health Management" (SHM) is perhaps a better description.

In the final analysis, the problem before us is to determine the techniques, processes, and technologies which allow a *system* to be built and operated in a dependable manner. As an example, the National Launch System (NLS) consists of vehicles, ground systems, and operations to provide launch vehicle services. It is not simply the launch vehicle itself. Specifically, as shown in **Figure 1.1-1**, there are four major elements involved with health management of a system. First, the system must be built in a manner which reduces the number of failures to a minimum. In other words, the design should remove as many possible system failure modes as possible, and be built using techniques and components which are as reliable as possible (within the cost constraints of the program). Second, the system should be tested and validated to find, and then remove any faults which may have occurred due to faulty workmanship, parts, or design. Third, should any faults slip through the testing, or occur during the operation of the system, fault prediction and tolerance techniques can be used to allow the system to continue to function normally, or to degrade in an appropriate manner. These include any compensation or reconfiguration of the system to ensure correct operation or safety of the system. Lastly, should a failure occur, there need to be mechanisms to isolate the cause of the failure, and feed the results of these fault analyses and experiences back into the design and verification of the system (or other systems) for continuous process improvement. All of these techniques and processes make up the *Health Management System* (HMS). The HMS is usually allocated across and integrated with all subsystems, as opposed to a separate subsystem. It is a system level function which may consist of functions distributed throughout various subsystems, and may or may not have its own "black box." Vehicle Health Management is that portion of the Health Management System which has to do with the health of the vehicle (as opposed to the health of the ground system which supports the vehicle).

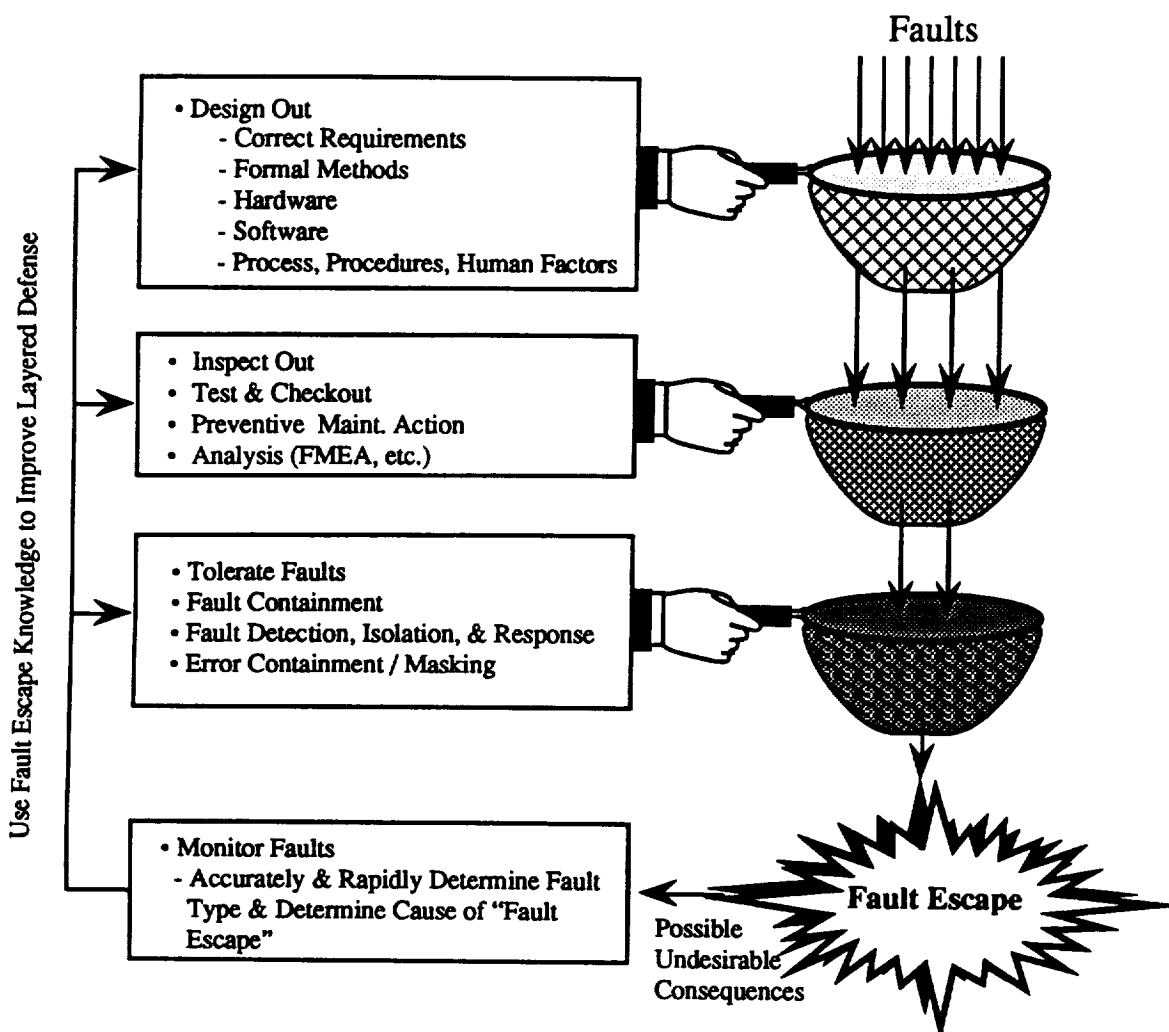


Figure 1.1-1 The Four Elements of the Health Management System

Underlying this document is the assumption that the primary difficulty in creating a dependable system is in understanding the complexity of the system, particularly in its response to faults, and then allocating design techniques appropriately to address those failures in a cost effective manner. *This can only be accomplished if the fault behavior of the system and the system design to deal with faults are built into the system from the start.* It cannot be accomplished as a band-aid after the system has been designed without considering failures. All of the elements of **Figure 1.1-1** must be designed into the system from the start. This document addresses the issues which face system designers when building and validating a dependable system, and presents a process for the System Health Management design within such a system. The implementation which results is the Health Management System for the given system.

The National Launch System (NLS) program is the initial target system for this document. However, the methodology discussed herein is generally applicable to systems of all types.

1.2. Scope

The first order of business, given the discussion of the problem in the previous section, is to define the terms which cover the spectrum of problems, processes, and technologies involved with building a dependable system. We shall start with some definitions needed before going further. The scope of System Health Management will then be explored and defined, and finally, the scope of this document.

1.2.1. Initial Definitions

System Health Management (SHM) is a term describing the “discipline” of health management for systems in general. It is analogous to a term such as “Propulsion”, which is used to describe the general field of propulsion systems or propulsion engineering. SHM consists of the processes, techniques, and technologies used to design, analyze, build, verify, and operate a system from the viewpoint of preventing or minimizing the effects of failure.

The *Health Management System* (HMS) is defined as the set of hardware, software, and operations that are implemented for a system to deal with faults. It includes all aspects of the implemented health management, at system, subsystem, and lower levels for a particular system. It is usually implemented as a set of techniques embedded within various subsystems, as opposed to a separate entity.

A *dependable system* is a system which performs its intended function when called upon. System Health Management can be used to achieve the goal of building a dependable system.

A *fault* is the physical or logical cause of an error. An equivalent definition is: “A deviation from desired or expected behavior which may manifest itself as an error.” In both definitions, the fault is the prior event which results in some “fault symptom.”

An *error* is a detectable undesired state. (It exists either at the boundary or at an internal point in the resource, and may be experienced by the user as a failure when it is propagated to and manifested at the boundary) Faults manifest themselves with detectable symptoms, which is the error. Sometimes the word *symptom* itself will be used. The word *anomaly* is also commonly used.

A *failure* is a loss of intended service that is suffered by the user. (designer's or user's intent). The system no longer performs its intended function.

1.2.2. Scope of System Health Management

The domain of System Health Management includes all causes of failure. Thus it encompasses not just random hardware faults, but design flaws, manufacturing problems, and human errors. The reason for this broad definition of SHM is discussed below.

The problem, as stated above, is to build a system which does not fail, or said in the reverse way, is dependable. To be more specific, as many faults should be prevented or avoided whenever possible, and tolerated otherwise. Since it is impossible to prevent all faults, fault tolerance is a necessary attribute of a dependable system. In the worst case, should system failure occur, enough information regarding the failure should be made available so that the problem which caused it can be fixed. Thus the first question which must be answered to determine the scope of SHM is to determine the causes of system failure.

The usual assumption is that random hardware faults are the primary cause of system failure. It is clear that these are valid faults, but it is incorrect to assume that these are the only, or even the primary causes of system failure. If the fault domain of the system is arbitrarily reduced to the space of random hardware faults, then the analyses and estimates of system dependability are likely to be significantly overestimated.

The actual causes of system failure vary greatly from system to system, depending upon mission length, criticality, complexity, repairability, and a host of other issues. For deep space missions, hardware faults are indeed significant, but are in fact very seldom random. They are almost inevitably traceable to specific flaws in the design or manufacture of the component. Equally significant for these missions are errors in the mission operation. For a vehicle with a two to ten year mission, the expertise required to understand all of the design features and nuances is very difficult to maintain. This is due to experts leaving the project, and also due to the fact that over such a long span of time, people forget what they once knew. Thus the Health Management System must be concerned with both of these failure types. As cases in point, the Russian Phobos spacecraft was lost due to bad command sequences sent to the spacecraft, whereas the American Voyager, Galileo, and Magellan missions have survived numerous commanding errors. This is largely due to on-board VHM (referred to as Fault Protection for these spacecraft) on the American spacecraft which is significantly more robust than their Russian equivalents. Since commanding failures often appear to the system as if they were hardware failures (the fault protection software often detects behavior which is unexpected from a mission standpoint), the last line of defense for saving the system from commanding errors is the on-board VHM.

For unmanned expendable launch vehicles such as Atlas, Titan, or Delta, the most common failure modes are traceable to manufacturing flaws. Thus the health management for these systems should concentrate on eliminating these flaws prior to flight. For a one-of-a-kind system, the design flaw is a much more significant system failure risk than a multiple-copy operational system. The fault set must include operational faults, if that is of concern for the system. It must contain transient failures, if Single Event Upsets or noise in the system can cause problems. The design flaw must be considered, if a generic or common-mode fault is a valid concern for man-rating or other issues.

Another important question is whether Health Management (HM) should be implemented at the system or subsystem level. It is quite true that subsystem engineers are the experts in their own subsystems, and that failure cases are considered and accounted for. A typical implementation for a HM system is that there exist HM functions at various levels all the way from components to the system level. Normally, as many of the HM or fault tolerance algorithms and functions are allocated to as low a level as possible. This makes sense for a number of reasons, not least of which is that the design engineers are the most capable of understanding and implementing HM techniques within their own subsystems or components. However, HM at these levels is not usually capable of handling all failure mode effects or responses by itself. Nor are the subsystem engineers in a position to assess questions of "balance" of HM capabilities within the system as a whole, where one subsystem could potentially have a very thorough HM scheme, and another subsystem could have very little. A typical example where both system and subsystem effects are present is an engine failure for a launch vehicle which has engine-out capability. The engine subsystem can handle its own fault detection and engine shut down in an independent fashion. However, it is also required to signal the vehicle so that the control algorithms can be updated to control with one less engine, and the guidance algorithms to plan a new trajectory to reach orbit. The typical HMS implementation is thus distributed in a manner appropriate to the functionality of the various subsystems. Similar considerations are used for distribution of the ground based HM. Thus HM is both a system and subsystem issue, where allocation of HM functions and tasks occurs on the basis of need. It is a misconception that SHM results in a whole new set of engineers at the system level ordering subsystem engineers to do that which they already know needs to be done. The actual situation is one where there is a person or small group at the systems level coordinating all of the vehicle health functions, working with appropriate personnel who are designing in the HM features of their subsystems and components, and building the system HM to deal with faults which have system level effects and interactions.

To summarize, the scope of SHM encompasses all processes, techniques and technologies which are used to make a system dependable. All causes of failure must be accounted for, not just the simple ones. It filters into the design of the overall system in terms of a coherent system design methodology, in terms of subsystem specific techniques and technologies to detect and tolerate failures, into component reliability and margin issues, and issues related to the cost of system failure. SHM is generally implemented by being embedded in the various components and subsystems of the system, as opposed to being a separate physical entity.

1.2.3. Scope of the SHM Methodology Document

It is appropriate at this point to define a new term to signify the process which this document will elaborate. The term “*Dependable Systems Engineering*” (DSE) is defined as the set of processes and techniques used to design, analyze, build, verify, and operate a system from the viewpoint of preventing or minimizing the effects of failures. Note that this definition is simply the definition of System Health Management, with “technologies” missing. (See Figure 1.1-2.) That is to say, DSE is the process portion of SHM, or in other words, is equivalent to the “SHM Design Methodology.”

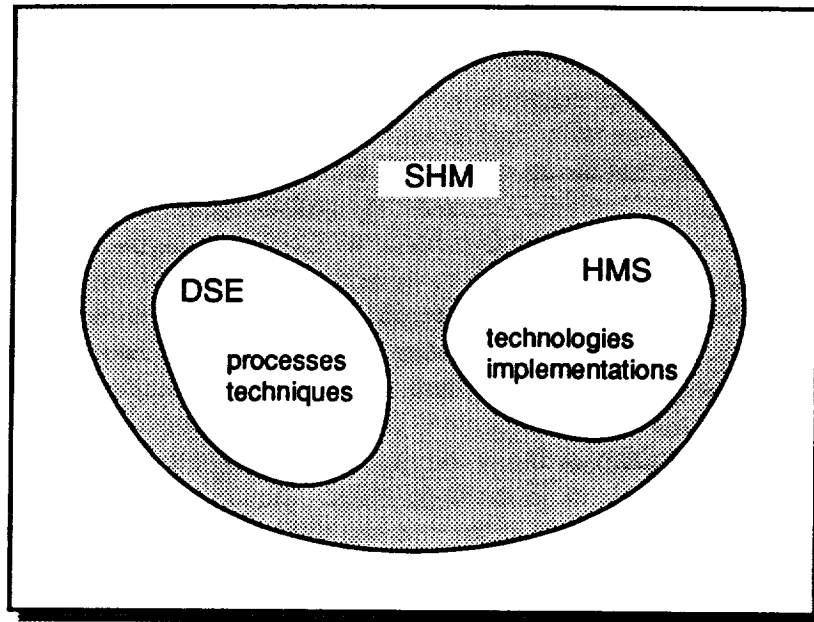


Figure 1.1-2. HMS and DSE are Subsets of SHM

Performing DSE tasks will result in a Health Management System, which includes the vehicle and the system as a whole. These include removal of possible failure modes early in the design process, test and rigorous verification to remove design flaws, automated or manual test of the system to eliminate process faults, fault containment and fault tolerance techniques so the system can survive a fault during its normal operation, and analysis techniques to analyze failures on the ground should they occur. Implementation of a particular HMS can include hardware, software, or operations, any of which can be either ground or vehicle based (the Shuttle has vehicle based operations using astronauts, for example).

The scope of this document is the process for designing a dependable system (i.e. DSE), using the broad definition of SHM, with an emphasis on space launch systems. Techniques and technologies used within particular subsystems (for example, Byzantine algorithms for computing systems or plume spectroscopy for propulsion) will not be discussed, except as examples to help illustrate the overall design process.

1.3. Objective

This document shall provide the basis for a design process for the discipline of System Health Management, and in particular shall emphasize the initial phases of a design process which can be used by the National Launch System program to design its Health Management System, and optimize its dependability. The primary objectives of this document are to specify the underlying principles which can be used to design a dependable system, and lay out a design process based on these principles. It is not intended as a "cookbook" which specifies every detail which a project must follow, but rather lays out the basic concepts and processes from which such a standard handbook can later be generated. The state of the art in System Health Management is such that all of the specifics required for the "cookbook" are not yet known. This document emphasizes exposition of the reasoning behind the steps which need to be taken to generate a dependable system. Although this approach for the document makes for longer reading, it was felt that a bare summary of the steps to be taken would be misleading. There are usually several approaches to generate any particular product in the design process, and the designer is going to have to use common sense to determine how to proceed for the system in question. Appendix D provides a short summary of the products which need to be generated at each phase of the design process.

1.4 Structure of this Document

Section 1 contains introductory material. Section 2 contains the "philosophical underpinnings" of the document, and also contains the assumptions which are applicable throughout the remainder of the document. The heart of the design methodology is contained in Section 3. At the beginning each design phase presented in Section 3 is contained a summary of the products and activities which occur within that design phase. Section 4 contains some recommendations for further work in the SHM field based upon the design process contained herein. Finally, the document contains four Appendices: References, List of Acronyms, Definitions and Methodology Summary.

—

.

.

—

.

.

—

2.0. System Health Management Issues and Approach

This section describes the overall approach to System Health Management underlying the methodology discussed in Section 3. It also discusses the applicability of systems engineering and concurrent engineering techniques within the SHM design process, and the variations to the process which are needed given different types of applications.

2.1 SHM Design Approach

This section will describe the constraints and technical difficulties involved with designing a system which is dependable, and the basic strategy for how these difficulties can be overcome. The strategy described herein underlies the entire SHM Design Methodology.

2.1.1. Infinity of the Fault Domain

In the design of a system which has strict reliability or fault tolerance requirements, there comes a point in time in the design process where the design engineers are faced with the task of proving the reliability or fault tolerance of the system. This task, as anyone who has worked it can attest, is extremely difficult. The reasons for this difficulty have to do not with complex calculations, but rather with the uncertainty in the engineer's mind as to whether *all of the fault possibilities have been considered*. Or, if the engineer or project has a quantitative bent, there is a related headache, for the engineer knows that *the numbers upon which the reliability calculations are based are unprovable, and are quite possibly wrong*. These two difficulties are related to the fact that the domain of faults for a system of any complexity is unknowable, and possibly infinite.

Despite these difficulties, which will be described in more detail below, the engineer must nonetheless design and build a system. Even though there are many things which are not known relating to the frequency and type of failures which the system will be subject to, the designer from experience does know quite a bit about the proposed system. The design methodology described in this document, and the basic philosophy under girding it, are based upon using the knowledge which the designer has, and creating the information which is needed to aid the designer in building a dependable system. To create such a methodology, the first step is to understand the difficulties involved.

One typical approach to analysis of a system's fault behavior is to use the Failure Modes and Effects Analysis (FMEA). This type of analysis looks at each component of the system, determines its primary failure modes and symptoms, and reports these failures and symptoms in a document. These failure symptoms can then be analyzed to determine their impact on the rest of the system. At the lowest level, the major difficulty is determining whether all of the appropriate failure modes of the component have been analyzed. Typical assumptions for electronic components include analyzing the effect of open and closed circuits on the local circuitry around the given component. Often not considered are intermittent failures, such as a heat related short circuit or bonding problem. Also, if the system in question has thousands or millions of components, simplifying assumptions must be used to reduce the analysis to a tractable problem. These include reduction of the types of failures considered, or looking not at single electronic components, but sets of electronic components. Similar considerations are applicable to non-electronic components.

To complicate the matter further, the effect that a given component failure has on the system depends upon the function the system is performing at the moment. Some failures have no effect at the time of their occurrence, because the system does not require the use of the component in question until later. Then, when the system uses it, the failure appears, even though it had been latent in the system for some period of time. This situation can cause a "double-failure" problem if the latent failure is in a backup component only used when the primary system fails. The primary unit fails, and switches to a redundant unit which has a latent failure, thus possibly causing failure of the entire system. Another example is a failure of a launch vehicle component when the vehicle is in test on the ground. At this time it has a much different effect than when the failure occurs in flight. It is quite possible for the failure to be undetectable on the ground, and then become visible and critical in flight. Thus every fault scenario must consider the different modes of operation of the vehicle to determine the effect of the fault. Lastly, the emphasis of FMEAs is frequently to concentrate on pushing the analysis to ever lower and more exhaustive levels of parts failure analysis, while overlooking interaction and timing variation effects at the interfaces. Ultimately, it is the effect of failures at interfaces which determines the criticality of the failure, and which must be analyzed in detail. The behavior and interaction of fault symptoms with the rest of the system is critical.

Switching to the quantitative mode of thinking, these same types of considerations corrupt quantitative estimates of reliability. Normally, the system reliability is calculated based upon estimates of random part failure from MIL-STD-217. All of the individual components are "added up" to determine the overall reliability of the system. However, experience has shown that random part failures account for only a small percentage of actual system failures. Almost invariably, system failure is due to failures in the manufacturing or design process. There is a growing consensus in industry that random part failure is a fiction, and that the physical or operational causes of failure must be sought out and fixed. Unfortunately, reliability estimates are based upon the most unlikely of all failure causes, a random event. That this leads to overly optimistic reliability estimates should not be surprising.

For a fault tolerant system, this problem is made even worse by the fact that in order to estimate the reliability of the system, estimates of fault detection and response coverage must be given. Given some failure, the designer must estimate the probability that the system will detect and respond correctly to the failure. This exercise is fraught with numerous opportunities for error. By definition, response coverage is the number of faults detected and correctly responded to divided by the total number of faults which could occur. (See Figure 2.1-1.) In order to calculate this number, it is necessary to determine the probability of each failure type, which can potentially change over time based on part life and system environment. Once this is known for all components, then all component failure rates are combined in the appropriate manner (based upon the specifics of the design).

The problem in applying this to a real system is that: First, the designer does not know the actual failure rates of the components, since there is typically little or no data on the reliability of the process which creates the components, which is a likely cause of component failure. Second, the analysis of the physical causes of failure based upon FMEAs is incomplete, since it is usually unlikely that all failure mechanisms have been thought of. Third, the probability of these particular failures is also unknown, making the system reliability calculation highly unreliable (pun intended). Fourth, the effect of failures upon the system potentially varies over time, depending upon the function the system is performing, thus complicating the analysis. Last, since it is likely the designer's detection and recovery scheme involves both hardware and software, an estimate of the efficiency of the algorithm, as well as the probability of a design flaw in the software itself needs to be made. Given all of these uncertainties and unknowns, any estimate can be challenged on valid technical grounds.

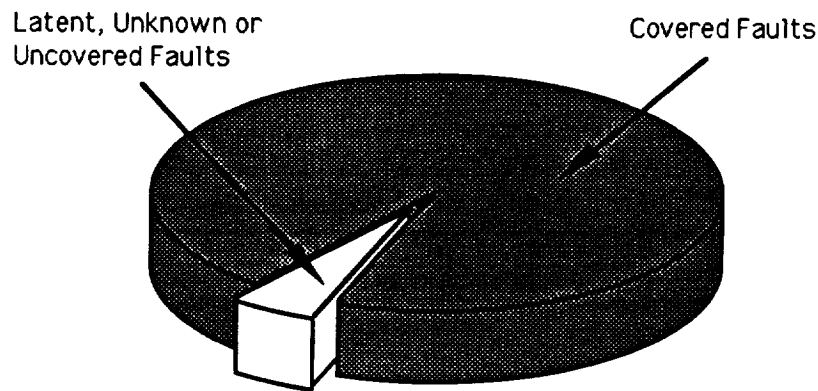


Figure 2.1-1 Fault Coverage

The only way to know how reliable a system will be is to operate the system under normal conditions. However, for a highly reliable system, this is not feasible. For commercial aircraft, with a 10^{-9} failure probability per hour requirement, it is literally impossible to test the system for the millions of hours necessary to prove its operation. For launch vehicles, which have a much lower reliability requirement (based solely on the fact that you cannot achieve the higher reliability goal), it is still not feasible to test, for each launch vehicle itself is prohibitively expensive, and is, except for the Shuttle, expendable. Thus quantitative testing is ruled out.

Qualitative testing is not much better. In theory every possible fault must be injected into the system. However, it should be clear by now that this too, is not practically possible. How does one know that one has thought of all of the fault cases? How do you know when it should be injected, for its effect varies dependent upon the mission phase? Even worse, not all faults can be injected into the system for very practical reasons, such as not jeopardizing this very expensive system by injecting faults into it. Thus the fault injection must be into a simulation, which may or may not have sufficient fidelity to mimic the real system.

Thus we are left with a dilemma. We cannot test for the reliability which is required. We cannot inject all of the faults. Reliability cannot be estimated with any certainty, since there is insufficient data available, compounded with the fact that what data does exist does not address the primary causes of failure to begin with. We cannot even assure ourselves that all failure modes have been thought of, or that their effect on all different mission modes has been considered. It seems there is no solution.

Fortunately, things aren't quite as bad as these gloomy statements would make it appear. Systems of all sorts work quite well most of the time, and the engineers who build them do have a good "feel" for the reliability of the systems they build, and what types of problems are most likely, even if they cannot pin down every last failure mode or give the exact probability of failure. Despite the imperfections of quantitative information, it is still of great value. Quantitative information still allows prioritization of efforts, to focus resources on the areas most likely to reduce failure frequency and effect. The key is using what is known, without ignoring the fact there are tremendous problems in trying to achieve exactitude.

The process described in this document is based upon knowledge of the uncertainties involved in trying to validate the dependability of a system. Since it is not possible to prove the reliability or fault tolerance of a system quantitatively, quantitative measures cannot be used exclusively as a validation criterion. However, there is experience with systems, which does enable an engineer to make reasonable estimates of a system's capability. This experience is certainly good enough to make relative assessments of one design versus another. That, after all, is what the design process is all about: trying to select and create the appropriate design for a given set of requirements. The key to the System Health Management design process is to provide to the engineers who are building the system the information they need to make design decisions with regard to the reliability, fault tolerance, and fault behavioral characteristics of their portion of the system.

2.1.2. Quantitative Criteria

During the initial design of a system (as opposed to upgrading an existing system), quantitative reliability or availability data is usually scarce or non-existent. This remains true through the entire design, build, and initial operations of the system. Thus a valid question to ask is, what is the validity or use of quantitative reliability estimates for the design phase of a project, if the data is nonexistent or unsubstantiated? How can quantitative initial requirements be formulated?

Regardless of the existence of data, it is still true that the customer will desire certain quantitative attributes of the system, for example, a 98% reliability or a 95% availability to launch on a particular day. These desires are true, whether or not they can be proven to exist in the system to be built. In addition, despite the lack of data, it is seldom if ever true that there is *no* data. Engineers with experience with the type of system in question will have some feel for the quantitative characteristics of the system, based upon their knowledge of the underlying technologies and processes. This can be translated, if one so desires, into estimates of reliability or availability, for example. Even though the specific numbers chosen cannot be justified, it can be reasonably argued that these numbers have the correct order of magnitude. It is also possible to compare different design options, knowing that relatively speaking, one design option may be more reliable or available than another, again based upon the knowledge of the underlying components (HW, SW or operational) of the system.

Quantitative specifications determine the 'order of magnitude' of the dependability problem to be met. For example, in the commercial airline world, very stringent quantitative reliability and availability requirements are set, to minimize the chance of catastrophic flight failure, and to simultaneously meet the airline's network schedule demands. These requirements force the aircraft designers to very comprehensive fault tolerance and health management strategies, from on-board fault tolerance which includes dissimilar design, to comprehensive maintenance plans which rely on both on-board and ground based fault prediction, detection, and isolation techniques. For a spacecraft, the requirements are generally more lenient in terms of availability, with correspondingly different strategies for fault tolerance and operations. Thus quantitative requirements set the basic framework of the design in terms of its technology and operation. However proving that the design actually meets a given specific quantitative reliability or availability number is virtually, if not actually, impossible.

In the commercial aircraft world, proving an aircraft design actually meets these stringent specifications has been an ongoing problem. To date, even with extensive testing and flight experience, the airlines, airframe designers, and certification agencies have yet to find an adequate solution to the problem of verifying and validating the dependability of the system. This has not prevented aircraft from being designed or certified, but it has certainly made it very difficult. The reality of the situation has been that the aircraft manufacturers do everything in their power to make the system both reliable and available (in large degree by using fault tolerance techniques), and do as much testing as they can afford to validate its safety. The certifying agency verifies these activities have actually been accomplished.

For spacecraft and space launch systems, similar considerations apply. Given the limited numbers of spacecraft and launch vehicles, there exists far less data on which to base quantitative estimates. The reliability and availability requirements are less stringent, but the limited numbers and high costs of failure make the space vehicle validation problem equally important, and equally difficult to achieve as the commercial airline counterpart. Since the cost of failure is high, extensive measures to make the system dependable are called for. However, due to the limited numbers of these systems, data regarding the reliability of the systems are extremely difficult to come by, thus making quantitative estimation a matter of guesswork.

It has been seen that quantitative requirements set an absolute measure which translates into certain types of design techniques appropriate to these quantitative goals. However, it is not possible to validate whether the particular goal has been achieved. This fact does not prevent the use of quantitative requirements, for they do express a customer desire, and they do in fact lead to specific design decisions *based upon the experience of the system designers*. This is a key point. Recognizing that the quantitative performance of the system from a SHM viewpoint cannot absolutely be determined, the designer does realize the relationship of certain design techniques with certain performance measures, and makes design decisions accordingly. For a spacecraft, a dual prime-backup system with cross strapping and safing routines has proven appropriate to long life missions with moderate availability requirements. A strict flight reliability requirement for launch vehicle translates into fault masking, but does not necessarily require dissimilar redundancy unless man rating is involved. A commercial airliner requires dissimilar redundancy due to its extraordinarily stringent reliability and availability requirements. The designer understands which techniques yield a certain range of reliability. However, since the exact quantitative figures for the system (whether it is maintainability, reliability, etc.) cannot be pinpointed, *the best that can be done is to choose or create a design which can be shown by analysis to yield the correct order of magnitude of the SHM quantitative measure in question.*

One way to use the quantitative data for design selection is to use worst case assumptions. Since the quantitative data are always debatable, a prudent procedure is to estimate a worst case reliability for each component, and use this figure as the basis for determining the technologies and architectures to be used. This can define a "worst case design" which accounts for the uncertainties in the data.

In addition, past experience does help pinpoint where problems are likely to be. A case in point is in launch vehicles. From 1966 to 1987, there were 742 American launch vehicle flights, with 58 failures (Figure 2.1-2). This data can help the designer in determining what sorts of systems have historically caused trouble, even if the next launch vehicle will not be exactly the same. For launch systems, propulsion is the primary area of concern. Avionic components and separation devices are the other areas which have caused failures. This information helps in the allocation of reliability to different subsystems, and helps determine where system health management techniques are likely to have the highest payoff. However, it does not aid in the validation of a new launch vehicle system in the sense of being able to use the data to prove with certainty a reliability or availability goal has actually been met.

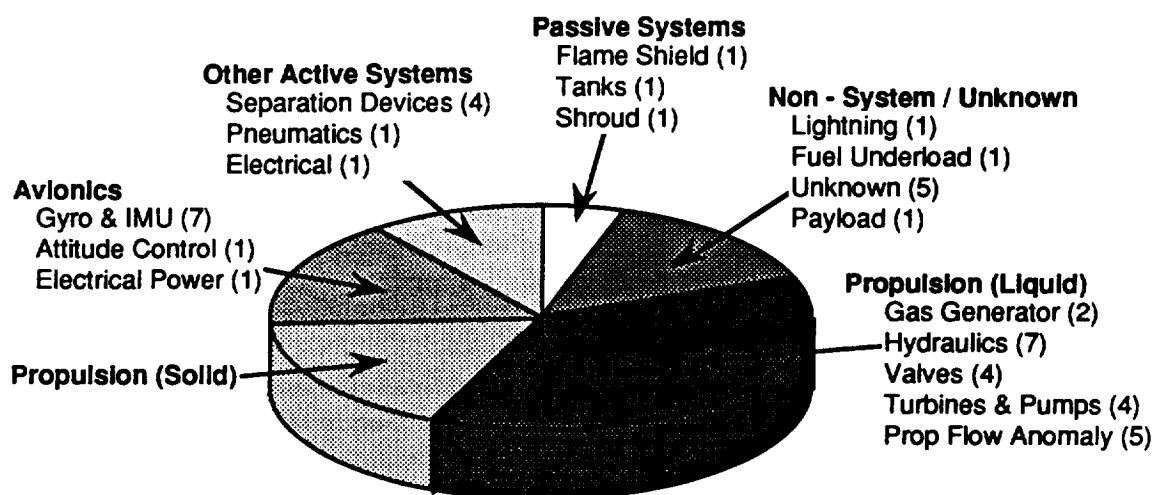


Figure 2.1-2. Subsystem Sources of Failure

A second use of quantitative measures is to assess the relative merit of one design versus another. Even though the designer cannot determine the absolute number with precision, it is often possible to know the relative value of one design versus another for a given parameter. This, again, is generally based on experience, although in this case, quantitative analysis of given configurations can yield useful results. Thus as a trade study tool, quantitative measures have a useful and practical role. As long as the assumptions for the input data are consistent and roughly correct, the trade study of one design versus another will be correct. When there is doubt as to the validity of a given input parameter, then it is possible to vary the input over the likely range, to determine sensitivity of the analysis to this uncertainty. In many cases the uncertainty will not be a major factor.

A third use of quantitative data is to “balance” the HM design. For a large scale system, the HM design features are embedded within various subsystems and components of the system. A typical problem is to determine how much HM needs to reside in each subsystem, and to make sure this is consistent with the design of other subsystems. As an example, is it sensible to use 12 sensors for the inertial measurement, 20 for propulsion, and 200 for the structure? If a designer of one subsystem specifies many sensors and algorithms monitoring many items, and a designer of another subsystem opts for a minimal set, how are the respective requests to be judged? Use of quantitative figures of merit can greatly assist in this problem, by basing the decision on explicit, quantitative criteria of cost, reliability, and whatever other major factors are appropriate for the items in question.

A last use of quantitative criteria is in analysis of specific problems in the design. As an example, a typical use of quantitative reliability estimates is when it is determined there is a single point of failure in a given design. Structural components are inevitably single points of failure, and the accepted practice is to assess the probability of failure for these components, which looks into lifetime issues, design margins, and other “fault avoidance” issues. If the chance of failure of the component is deemed low enough, then the single point failure requirement is waived for that particular case. This shows that even in systems where the primary requirements are qualitative (no single point of failure, for example), quantitative criteria are still used.

To summarize, there are four major uses of quantitative HM data: to determine the appropriate types and performance levels of the technologies to be used in the system, to assess different designs using the quantitative measure as a figure of merit, to balance the relative amount and type of HM within various subsystems, and in detailed analysis of particular design problems. Conspicuously absent is use of quantitative measures as validation criteria, except in particular instances. For this, qualitative criteria are necessary.

2.1.3. Qualitative Criteria

Qualitative requirements have played a major role in development, verification and validation of the space systems health management design. Typical requirements are “no single points of failure” or “fail-operational/fail-safe.” The primary reason why qualitative requirements have dominated is the lack of data relevant to HM of space systems. As noted in the previous section, there are relatively few space systems in existence, and the ones in existence now are significantly different from those flown in the past.

As an example, how relevant is it to use data from the Shuttle program for an Atlas launch vehicle? The Shuttle's systems are quite different from Atlas, and vice versa. Evaluating the difficulties with avionic or propulsion components for the Shuttle do not necessarily imply anything about the components used for Atlas. Even within a given launch vehicle family, there has been substantial evolution over time. The Titan IV launch vehicle is quite different in a number of ways from Titan II or Titan III. Is it relevant to compare a mechanical Inertial Measurement Unit technology from a Titan II to a ring laser gyro technology planned for an upgraded Titan IV? Similar considerations apply to spacecraft and upper stages.

Because of problems such as these, space system designers have tended to rely upon qualitative requirements to achieve their goals. Thus, the requirement for a probability of failure of 10^{-9} per flight hour for commercial aircraft critical systems translates perhaps into "fail-op, fail-op, fail-safe" for a man rated space vehicle. It turns out, when the design of the systems are compared, there are a number of similarities in the actual designs used in these very different systems to meet these stringent requirements, be they quantitative or qualitative. Both need dissimilar redundancy, both need substantial levels of fault tolerance to meet their requirements.

Qualitative requirements translate more directly into a design than do quantitative requirements. For example, levying "single fault tolerance" for a certain set of faults specifically implies redundancy of some sort, whereas a quantitative requirement could imply either a very high reliability single string design or fault tolerance. Some analysis work or assessment is required to determine the appropriate level of fault tolerance which corresponds to the quantitative requirement. This brings up the point that fault tolerance, properly speaking, is a qualitative attribute of a system. A design can tolerate a certain number of faults. That is its fault tolerance capability. This capability does in fact map into a quantitative figure, depending upon the design, the quality of parts in the system, and the effectiveness of the fault tolerance techniques used in the system. However, in and of itself, fault tolerance is not a quantitative feature.

The primary advantage of the qualitative fault tolerance requirement is that it is easier to verify that it is met in a particular design. From a design standpoint, if there is a requirement for single fault tolerance to a particular class of faults, for example, permanent and transient hardware failures, then the designer can build in the appropriate type and amount of redundancy to achieve that goal. Better still, the system can be analyzed using an FMEA, to look at all conceivable permanent and transient failures of the hardware. Any single component failure which causes failure of the entire system is in violation of the requirement, and must either be designed out, or the requirement must be waived in this instance, which can only be done with special approval.

Contrast this with the quantitative criteria. In the quantitative case, the designer is trying to prove the design can meet a certain numeric value. Presume a single component failure in the system is found which causes system loss. Due to the large uncertainties of the data upon which the analysis relies, the failure case could be argued to be either significant or not. Since any particular failure is quite unlikely in any event, the designer has an "out", as long as plausible evidence exists that the event is unlikely, it need not be fixed, nor necessarily even reported. Unless there is some system of finding and then reporting all single points of failure, and associating probabilities with them, the system as a whole could have many possible failure modes, all of which are justifiable given certain assumptions as to the input data. If this problem is to be alleviated, then all failure cases must be reported and collected, as in the qualitative case, and then argued individually from a probabilistic standpoint, thus ultimately allowing the systems engineers to determine how many should be fixed or not. This is basically the same situation as is required in the qualitative case, with the difference that the qualitative requirement guarantees the reporting mechanism, and the assumption is that a problem should be fixed until proven it need not be. In the quantitative case, the assumption is that the problem should *not* be fixed until it is proven it should. This is potentially asking for serious trouble with the system design.

In the verification and validation of the system, the qualitative approach also allows for a fairly straightforward approach to testing. Since the analysis is on a case by case basis, the testing simply uses specific failure cases as faults to be injected into the system to verify its ability to tolerate failures. This is not necessarily very easy, due to the large numbers of failure modes, but given some judicious selection of faults which will be injected into the system, it provides a basis for a coherent test program, and also a basis for designing fault injection capabilities into the system.

Summarizing, qualitative requirements are in the HM field more straightforward to use for design and validation. The designer can map the qualitative requirement directly into an explicit design, and this design can be tested both analytically and in the lab by fault injection techniques. These are done by FMEA, which "mentally" injects the failure and analyzes the effect, and direct injection of the failure into a simulation or into the as built system. The qualitative requirement is related to the quantitative requirement given a specific design using components with specified life and performance factors.

2.1.4. Summary

The methodology described in this document assumes that both quantitative and qualitative requirements and measures have a place in the design of a dependable system. Neither are completely sufficient by themselves. In a system which primarily uses quantitative measures, it is still necessary to perform FMEAs to determine the possible failure modes of the system and assess into their probabilities. In a system which uses qualitative measures, when a single component failure causes system failure, the failure mode must be assessed from a probabilistic view. The key is to use these two modes of design and analysis in the way which is most practical and effective.

The strength of the quantitative method is its ability to allow for comparison of differing designs and system attributes, and to assess the margins in fault avoidance techniques. The quantitative method for HM does not easily accommodate testing, verification, or validation.

Qualitative methods easily map into a specific design, and are straightforward to use in the V&V of a system. However, they do not easily allow for comparison of designs, for fault avoidance assessment (design margins), or assessment of the likelihood of a particular failure when a programmatic decision must be made as to whether to allocate the resources to fix a system failure mode.

Both techniques can be used to determine the type of design techniques which should be used for a given system. For a launch vehicle, specifying quantitative reliability and availability requirements ultimately yields an appropriate HM design if the proper analyses are done. Similarly, specifying qualitative fault tolerance and maintenance requirements can accomplish the same goal. Ideally both types of requirement are levied, so that unambiguous interpretation follows.

In whatever form the initial customer demands are given, ultimately both quantitative and qualitative requirements must be specified, to allow for appropriate design and validation of the system. The qualitative requirements determine the fault tolerance capability of the system, which then drives the HM V&V program. The quantitative requirements specify the target allocations of design margins, reliability, cost, and maintainability, which allow the designer to set appropriate figures of merit for assessing the design, whether it uses fault tolerance or fault avoidance techniques. The methodology in this document is based on determining the appropriate types of quantitative and qualitative requirements and analyses for design of a dependable system.

The basic assumptions of the methodology with respect to quantitative and qualitative requirements and techniques are:

- Qualitative requirements and techniques are used to specify the fault tolerance capability of the system, and are used as the primary means of system V&V.

- Quantitative requirements and techniques are used to assess the effectiveness of fault avoidance aspects of the design, to perform design comparisons using quantitative figures of merit, to balance HM design features, and to aid project decisions when qualitative requirements are violated.
- Both techniques are valid and necessary in the initial phases of the system design to determine the appropriate types of technologies and techniques to be used in the system.
- A summary of Qualitative and Quantitative criterion is shown in Table 2.1-1.

<ul style="list-style-type: none"> • Qualitative Criterion <ul style="list-style-type: none"> – The Qualitative Domain Is Used in Identifying Classes & Particular Faults for the System – Fault Tolerance Is a Qualitative Characteristic • Quantitative Criterion <ul style="list-style-type: none"> – Extremely Useful To Compare Different Design Concept & Provide Relative Ranking – Cannot Prove Specific Quantitative Criteria Have Been Met – Uncertain & Unavailable Reliability Data – Too Many Permutations & Combinations of Faults for Complete Testing

Table 2.1-1. Summary of Qualitative and Quantitative Criterion

2.2. System Process Assumptions

2.2.1. Role of SHM in Systems Engineering and Integration

As shown in Figure 2.2-1, the health management system design is an integral part of the total launch system design. Therefore, the SHM design methodology must be interwoven into the launch system systems engineering process.

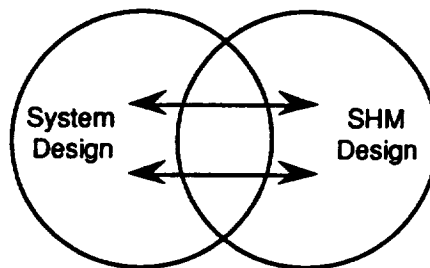


Figure 2.2-1. SHM/System Design Overlap

This methodology assumes the integration of the SHM effort into a larger systems engineering process is consistent with MIL-STD-499B (draft). The MIL-STD-499B systems engineering process follows the iterative path shown in **Figure 2.2-2**. The standard does not currently address SHM issues. This document supplements the information currently in MIL-STD-499B with SHM processes which can ultimately be part of the standard engineering process as represented in that document. It is not the intent of this document to describe all the details of the launch vehicle systems engineering process, but rather the SHM specific aspects of systems engineering.

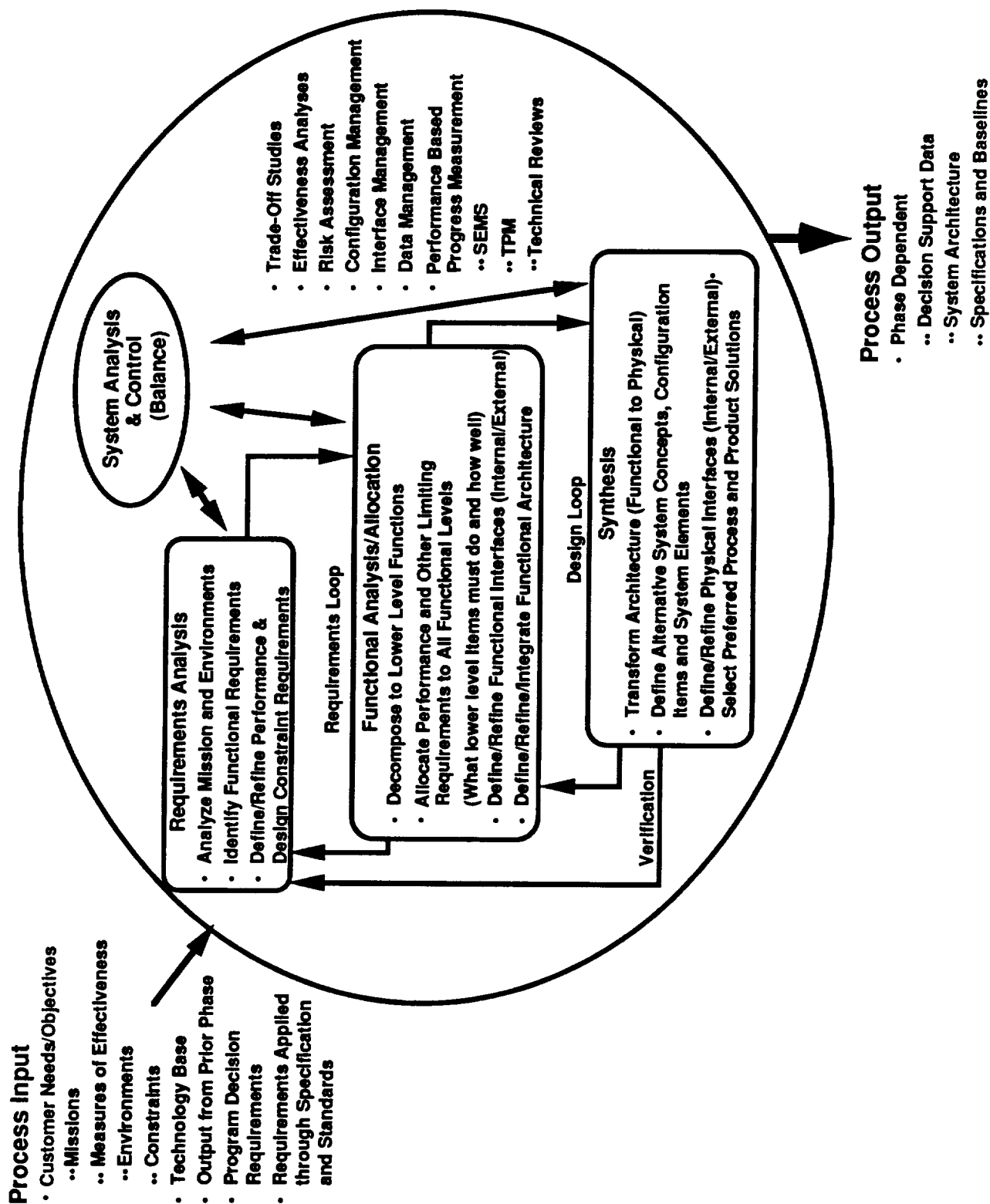


Figure 2.2-2. Systems Engineering Process

There is a continual iteration between SHM requirements development and the design implementation. The requirements development process begins at a high level of abstraction, and works to a level of better fidelity as the design process proceeds. Key to this methodology is development of SHM requirements at appropriate levels of fidelity and consistency during the different design phases which meet the needs of the designers of the health management system. Shown in **Figure 2.2-3** is a condensed version (only through preliminary design) of the iterative development of SHM requirements along with the initial phases of the SHM design implementation. The steps of **Figure 2.2-3** will be discussed in detail later in this document.

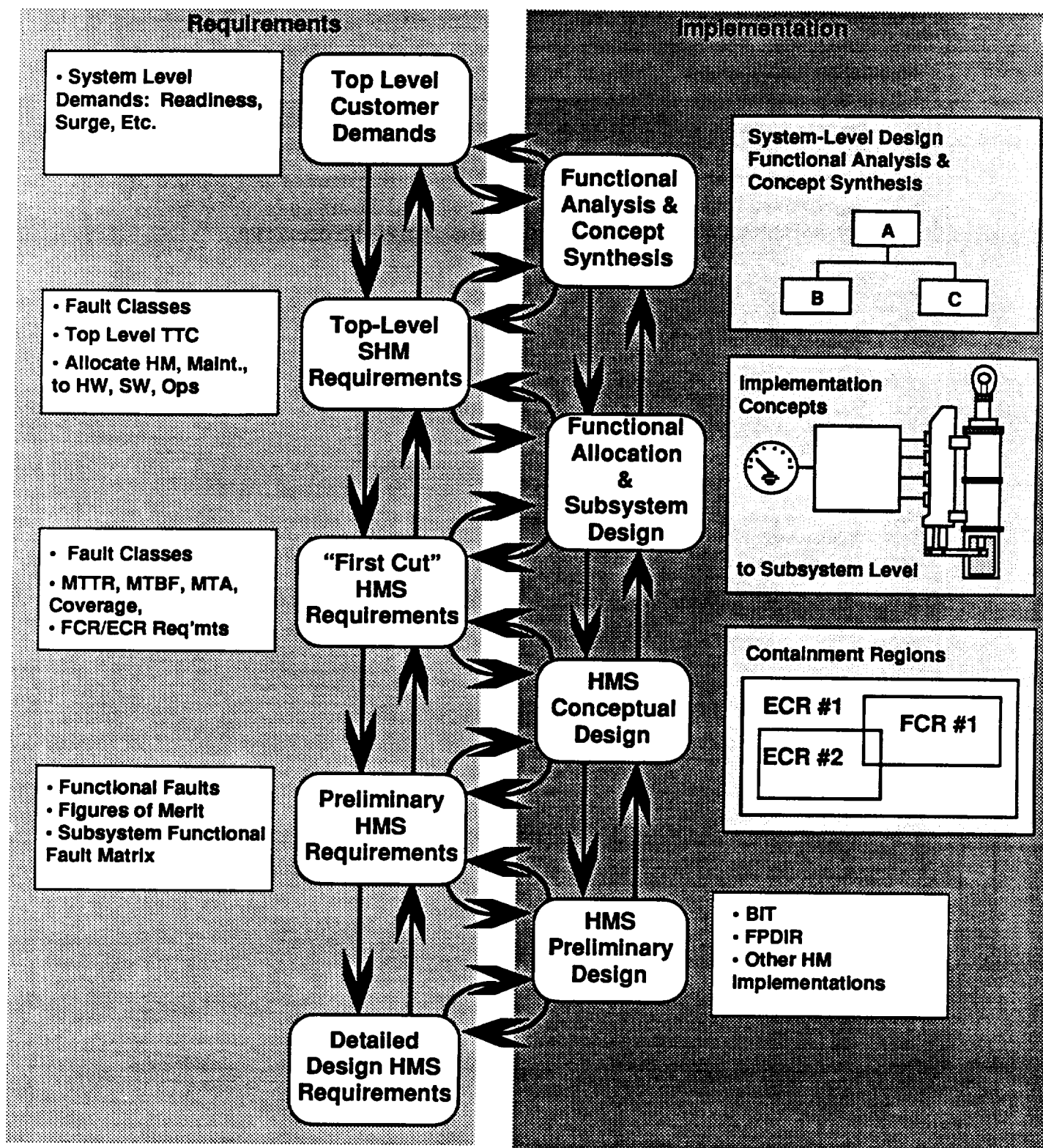


Figure 2.2-3. Requirements/Implementation Iteration

There is an intimate relationship between the SHM design and the overall system integration effort. This relationship is most easily explained by considering the specific tasks involved in the SHM design. The personnel tasked with designing the SHM must understand the behavior of all parts of the system under fault conditions. In order to do this, they must understand in great detail how each element of the system functions normally, how each element interacts with all other elements under normal condition, and finally, what changes and behaviors occur when fault conditions are injected into the system. This includes understanding of the mission phases and the different functions the system performs in these phases, since fault behavior changes under different operating conditions. For a launch vehicle, these phases include the ground test and checkout of the system during the system build and integration leading up to launch, as well as the flight. Since the objective of the ground checkout is to find failures or manufacturing errors, the fault behavior of the system in ground test is just as important as in flight. During the design phases, the SHM personnel levy requirements on the system to allocate redundancy and fault tolerance implementations, design the vehicle flight FDIR algorithms at the system level, as well as levy requirements on algorithms and sensors within subsystems. To perform this task, they build behavioral models of the system, and build or modify the full scale simulation to inject failures into the system, test out the FDIR algorithms, and assess vehicle tolerance to failures. The testing of the system design is a key issue, for the system fault tolerance and SHM features can only be tested by injection of failures into the system, thus leading to capabilities which must be built into simulations and ground test equipment. The checkout of the system is equally important, and the SHM personnel levy requirements on it. These requirements form the basis for the internal built in test (BIT) or ground based diagnostics. In addition, because of their intimate knowledge of the system's behavior in anomaly conditions, they become the team responsible for the contingency planning, the recovery operations, maintenance, post-test analysis in terms of looking for anomalies, and determining what responses to take. In short, the SHM team becomes the most knowledgeable single group with regard to the overall system operation, and effectively integrate the system because of this knowledge.

2.2.2. Integrated Product Teams/Concurrent Engineering

The methodology in this document presumes the use of Integrated Product Teams (IPTs). Integrated Product Teams are groups of designers, managers, and technicians which are formed to ensure the concerns of each group involved in the design, management, manufacturing, or operations are accounted for in the design of the system. Since system health management integrates into the entire systems engineering and integration process, SHM philosophy and training should be infused into the members of IPTs. This promotes SHM as a general systems engineering element, and in conformance with the philosophy of Integrated Product Development, includes SHM as an integral contributing part of design instead of a separate group monitoring the process. Since SHM is implemented at many levels, and is embedded in most subsystems and components, each member of the team needs to be aware of SHM issues and techniques which apply to their system.

Design reviews for projects utilizing this methodology should have SHM specific design questions to determine if SHM issues have been adequately represented in the Integrated Product Teams (IPTs). The danger of assumption of complete infusion of SHM into the IPTs could be that no team is adequately addressing SHM and bringing together its particular activities efficiently. This will be further discussed in the recommendations section.

2.2.3. Spiral and Waterfall Models

A difficulty faced in this methodology is assignment of SHM technical elements to distinct phases, and discussion of order of execution of SHM design steps. This methodology will require appropriate customization to the specifics of each project. As reflected in the iteration arrows of Figures 2.2-2 and 2.2-3, it is erroneous to neglect the feedback and several iterations between requirements, functional allocation, and design synthesis. The waterfall with iteration arrow can also be viewed as a design spiral, as shown in Figure 2.2-4, a depiction more vividly reflecting the iteration process.

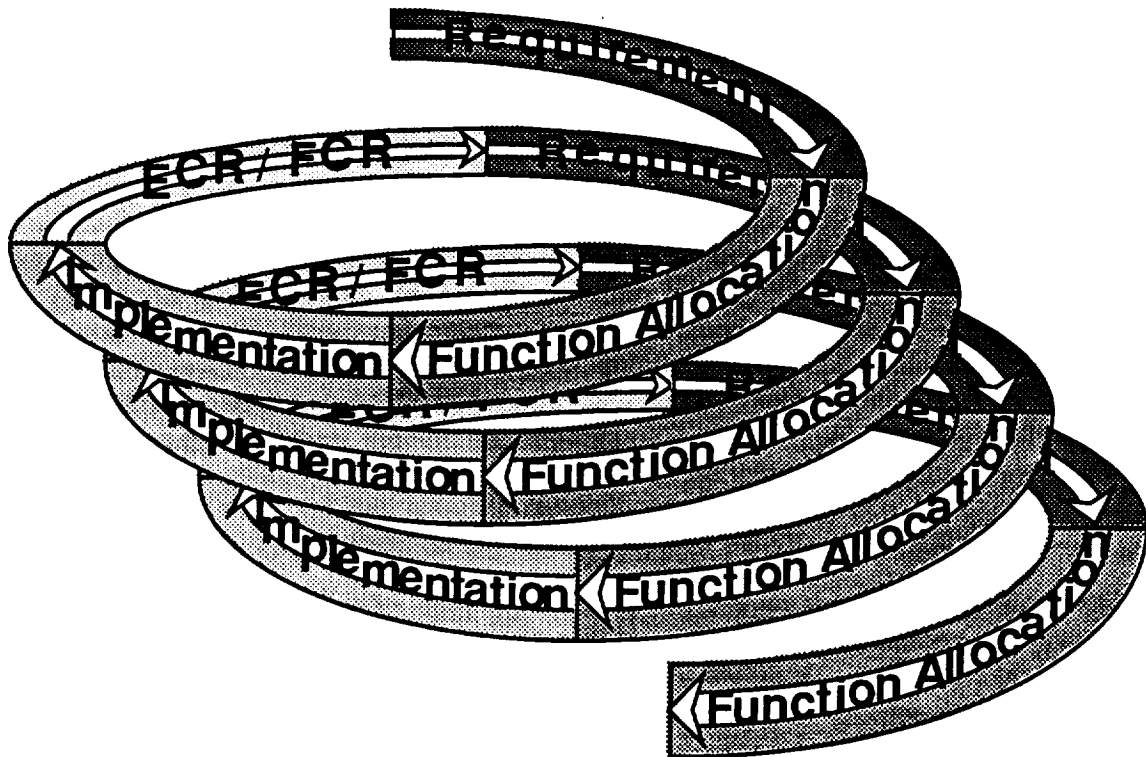


Figure 2.2-4. The Design Spiral

Contrasting with the spiral view of the design process is the typical “waterfall” view of the design process, as shown in Figure 2.2-5. In this view, each phase of the design leads logically to the next phase of more detailed and complete design. Due to the reality of having to create a “linear” document, the headings within this document will appear as the standard “waterfall” headings for design phases. As the system is developed, it is true that greater levels of detail are generated, and certain project reviews, such as the System Design Review, Preliminary Design Reviews, etc., must be completed. For simplicity, we have assumed the typical waterfall headings. However, the reality is that many feedback loops and design iterations occur as necessary, and the methodology assumes these are taking place, even though this fact is not always discussed at each point in the document.

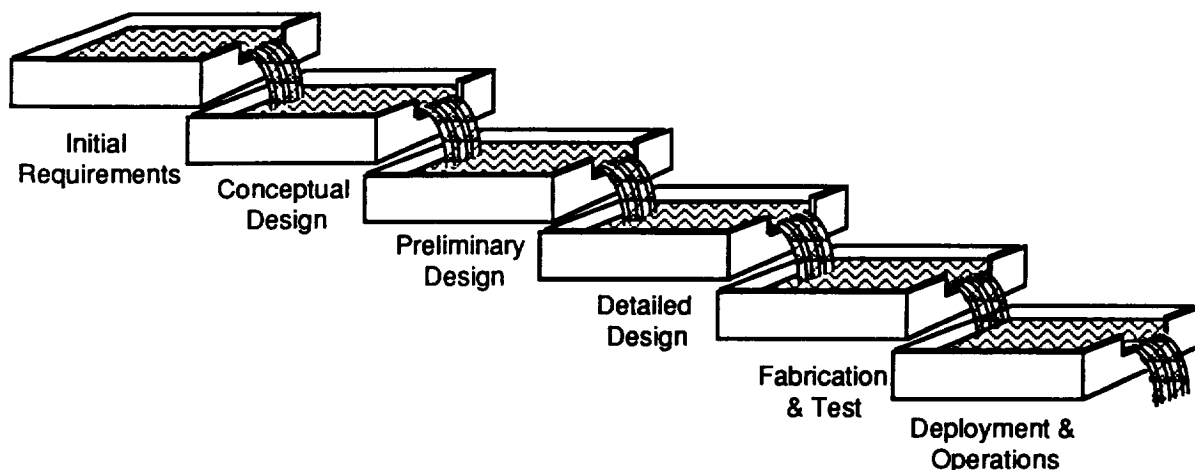


Figure 2.2-5. The Waterfall View of the Design Process

2.2.4. Rapid Prototyping

The typical systems engineering design process assumes that the system can be designed in a top-down fashion. Specifically, it often presumes that high level decisions are technically feasible at lower levels, or in detail design. This assumes that there are personnel with sufficient expertise or experience to understand the implications of a certain design decision, at least enough to understand whether anything precludes the design from being successful. This assumption is not always valid. On occasion, the information available is inadequate to make a design decision, for there are potential problems at lower levels of the design which may preclude a certain design option from being viable. Rapid prototyping allows for a quick determination of the feasibility of certain implementation concepts from a technical viewpoint. It needs to be used in conjunction with the top down process, in order to acquire information about the system when it does not exist from prior experience. Where the design timetable allows, much greater SHM design confidence and risk reduction can be provided by rapid prototyping.

2.3. Overview of the SHM Design Methodology

This section gives an overview of the SHM Design Methodology, introducing the basic concepts and flow of the SHM tasks within the overall design of a subsystem. **Figure 2.3-1 (a and b)** shows the 6 major time phases of the system design process across the top, and the major elements of the system process which evolve through time down the side. The matrix illustrates typical processes and products which occur through time associated with SHM topics and processes. **Sections 2.3.1 through 2.3.6** will discuss the major features of the methodology, and **section 2.3.7** will discuss the flow of these elements through the “waterfall” flow of the system design process (see **section 2.2.3**). In other words, **sections 2.3.1 through 2.3.6** will discuss the items on the side (the “elements” of the process) as they evolve through time, and **2.3.7** will concentrate on the basic products which are typically generated at each phase of the design process.

	Initial Requirements	Conceptual Design	Preliminary Design	Detail Design	Fabrication & Test	Deployment & Operations
Quantitative Requirements	<ul style="list-style-type: none"> • Availability • Reliability Allocation • Margin Philosophy 	<ul style="list-style-type: none"> • Maintainability Metrics, MTTR, etc. • Time to Criticality at System Level • Design Margin Establishment 	<ul style="list-style-type: none"> • Dependability Metrics Refinement • Time to Criticality at Subsystem Level • Margin Requirements 	<ul style="list-style-type: none"> • Final Margin & Reliability Requirements 	<ul style="list-style-type: none"> • Requirements Updates 	<ul style="list-style-type: none"> • Requirements Updates
Qualitative Requirements	<ul style="list-style-type: none"> • System Fault Tol. Reqmts. • System Fault Allocation Reqmts. • Fault Classes for Fault Tol. • Isolateability 	<ul style="list-style-type: none"> • Subsystem Fault Tol. Reqmts. • Subsystem Functional Fault Reqmts. 	<ul style="list-style-type: none"> • Fault Injection Reqmts. • Software & Hardware Operational Reqmts. 	<ul style="list-style-type: none"> • Final System & Subsystem Component Requirements 	<ul style="list-style-type: none"> • Requirements Updates 	<ul style="list-style-type: none"> • Requirements Updates
Fault Set Definition	<ul style="list-style-type: none"> • Fault Classes • Major Implementation • Fault Types (Engine Out, Electronics, Etc.) 	<ul style="list-style-type: none"> • Subsystem Functional Faults (Top Down) • Preliminary FMEA (Bottom Up) 	<ul style="list-style-type: none"> • Refinement of FMEA <ul style="list-style-type: none"> - Quantitative Failure Rates • Gathered Fault Combination Method (GFCM) ... Fault Set Reduction for Fault Injection 	<ul style="list-style-type: none"> • Final FMEA • Refine Fault Set • Reduction for Fault Injection 	<ul style="list-style-type: none"> • Updates to Fault Set 	<ul style="list-style-type: none"> • Updates to Fault Set

Figure 2.3-1a. Overview of SHM Task Part 1

	Initial Requirements	Conceptual Design	Preliminary Design	Detail Design	Fabrication & Test	Deployment & Operations
Fault Analysis & Modeling	<ul style="list-style-type: none"> System Cost & Reliability Trades 	<ul style="list-style-type: none"> System Interaction Time to Criticality Functional Fault Matrix Initial Behavioral Model 	<ul style="list-style-type: none"> False Alarm Analysis Detailed System Modeling Detailed Reliability Modeling Simulation With Fault Injection Cost/Reliability Analysis for Parameters 	<ul style="list-style-type: none"> Simulation With Fault Injection Detailed Alarm/Action Threshold Analysis Detailed False Alarm Analysis 	<ul style="list-style-type: none"> Fault Injection Into As Built System Alarm Threshold Testing With Integrated HW/SW System Characterization Model Updates 	<ul style="list-style-type: none"> System Characterization Model Updates Fault and Contingency Analyses
System Design	<ul style="list-style-type: none"> Initial System Concept Operation & Maintenance Concepts Health Mgmt. Data Flow Plan 	<ul style="list-style-type: none"> Initial Subsystem Concept ECR/FCR Definition at Subsystem Level First Cut at Parameter Set Degree of System Autonomy (Human Role) HM Software Plan Integration Design Provisions for Passive Fault Tol. 	<ul style="list-style-type: none"> Refinement of HM Architecture Concepts Detailed ECR/FCR Complete FPDIR Plan Parameter Refinement, Algorithm Dev. Sensor Selection 	<ul style="list-style-type: none"> Design Implementation Detailing HM Thresholds (Alarm, Reaction, Persistence, etc. Established) Detailed Data Mgmt. Plan HM Ground Eqmt. Design Refinement 	<ul style="list-style-type: none"> Design Feedback Threshold Adjustment from System Characterization 	<ul style="list-style-type: none"> System Characterization Design Updates Contingency Plans
Verification & Validation	—	<ul style="list-style-type: none"> V&V Plan Draft for SHM Allocation of V&V Methods: Test, Analysis, Proof, Simulation 	<ul style="list-style-type: none"> Incorporate Prelim. FMEA Into V&V Define Fault Injection Techniques Proof of Key Algorithms 	<ul style="list-style-type: none"> Test Procedures V&V By Analysis, Simulation, Test, & Formal Proof 	<ul style="list-style-type: none"> Subsystem and System Testing Under Stressing Conditions & Fault Conditions 	<ul style="list-style-type: none"> Testing Updates

Figure 2.3-1b. Overview of SHM Task Part 2

2.3.1 Quantitative Requirements

As discussed in section 2.2.2, quantitative requirements are used throughout the design process for performance of trade studies, assess performance of technologies, to balance the HM features among various subsystems, and to assess particular designs and problems found in the design. As the system design progresses, these requirements become more specific, detailed, and voluminous. Initially, the basic customer demands for the system are levied in terms of top level requirements such as “95% availability to launch on time”, and “98% ascent reliability.” These quantitative requirements are then allocated to different portions of the system. For example, the 95% availability to launch on time can be allocated to a weather related probability based on vehicle performance capabilities in adverse weather, and a ground system reliability. Similarly, the ascent reliability can be allocated in varying percentages to different subsystems. It should be noted that these allocations are goals in the early phases of the design, for insufficient detail exists to decide if the allocations are the best for the given system concept. Various trades must be performed to determine the optimal allocation.

The quantitative requirements which typically have the most influence on the SHM design are availability, reliability, cost, safety and fault timing (time-to-criticality). These requirements are driven to lower levels of detail through the design process, and “fan out” into a number of requirements on the reliability of subsystems, the cost of subsystems, performance and structural margins, maintainability and repairability, and system processing speed and timing.

Quantitative requirements are typical for military systems, but less common for NASA systems. They reflect certain types of analyses and trade studies which are performed for the system, which will be discussed below. A relatively new feature of a quantitative nature which this methodology calls out are the timing and speed requirements levied on subsystems to respond to faults. These requirements, related to “time-to-criticality” will be discussed in section 2.3.4, and throughout the methodology.

2.3.2 Qualitative Requirements

Qualitative requirements are needed primarily when quantitative information is lacking, and for verification and validation of the system. From the standpoint of SHM, the specification of the fault tolerance of the system is necessary. This means that the number and type of faults to be tolerated by the system, and the manner in which they must be tolerated, must be specified. As an example, in the initial requirements phase, the overall philosophy of the fault tolerance of the system is levied by specifying "single fault tolerance" to various fault types, such as hard and transient electronic faults, human physical performance faults, and software design faults. In addition, special faults to be tolerated can be called out, such as "engine-out" capability for a launch vehicle, which is critical to the design for various reasons. Terms such as "fail-safe" imply that the tolerance to a failure need not be full, but only partial in the sense that the system need only operate in a degraded mode after the fault occurs. Other requirements, such as "fault isolateability to the Line Replaceable Unit" (LRU) are also typical in early phases. In later phases, qualitative requirements are driven to greater levels of detail, and include requirements on the system for validation purposes such as fault injection capabilities which must be built into support equipment.

2.3.3 Fault Set Definition

A major element in the design process for SHM is the progressively more detailed determination of the faults which are associated with the system. At the very beginning, in the initial requirements and conceptual design phases, the actual implementation of the system is not yet known in detail, but yet the overall fault tolerance and fault avoidance requirements must be specified. These requirements are sensible only when levied against some specific sets or classes of faults. The requirement "fail-op, fail-safe" is incomplete without specifying what set of faults it is intended to protect against. Thus, this level of fault tolerance can be levied for electronic components, but not for structures, for transient failures but not design flaws. Specification of the fault tolerance in this initial phase determines the overall system fault tolerance philosophy. In the conceptual design phase, the concept for implementation of the system is determined. During this time, the top-down fault analysis process begins, by determining the consequences of failure of system and subsystem functions. The particular failure modes are not determined, but rather, if a function (for whatever reason) fails, how does the rest of the system respond, and in what period of time. For example, if the power source for the system fails, the entire system fails within milliseconds. If the telemetry system fails, the system is degraded, but still operates.

During the preliminary design phase, the system is defined sufficiently to perform a preliminary Failure Modes and Effects Analysis (FMEA), a first cut at a “bottom-up” failure analysis. At this time, specific failure modes of components (usually at the box level) are postulated, and their effect on the system determined. This “bottom-up” analysis is used to analyze the design, and also as input to the V&V plan, for specific faults must be injected into the system in order to test it. Only when the final design is completed can the final FMEA be completed. At this time, any additions or changes to the fault set are made, and the appropriate changes to testing are determined based upon these changes.

2.3.4 Fault Analysis and Modeling

Throughout the design process, and continuing through operations, the system is analyzed and modeled to determine its behavior under fault conditions. In addition, the cost and performance effectiveness of various design techniques are determined, requiring various types of analyses and models. Initially, when various system concepts are being traded against each other, cost, performance, and reliability analyses are performed to aid these decisions. Once these trades are completed, analysis of the selected system concept or concepts continues, looking in further detail at the concept feasibility. From the standpoint of SHM, the “Time-to-Criticality” (TTC) analysis is key. This analysis, at the top level, investigates the amount of time between the failure of a function and the adverse event or events which occur to the system based on that failure. As noted in section 2.3.3, the loss of a power system brings system failure typically within milliseconds, whereas loss of telemetry does not bring system failure at all (i.e. a TTC of infinity). This timing analysis is very important at this time, for if the system is built to recover from or report failures, the time in which that action must successfully function must be accommodated by design. This time to respond to failures becomes a requirement levied upon various subsystems, driving various performance aspects of the design.

At the next level, once the subsystem concept has been determined, a functional fault analysis occurs. This analysis takes a first look at the implementation to determine if the subsystem meets the requirement levied by the TTC analysis. In addition, other information is generated, including the fault propagation behavior of functional faults, and the effect of these faults upon the rest of the system. During this phase, reliability estimation of the system is performed based upon historical knowledge of the reliability of functions and subsystems for the application the system is designed for. These initial estimates provide a rough estimate of reliability of the system, to determine if the system can meet the goals laid upon the system.

During the preliminary design phase, the detailed simulations of the system are built. In order to support SHM, these simulations must be built with fault injection capability right from the start. For example, in order to test a fault detection algorithm embedded in the control system, faults within various sensors, actuators, I/O devices, and processing components must be produced in order to stimulate the algorithm. Another major activity during this phase is the analysis of the cost effectiveness of monitoring various parameters within the system. If a particular failure mode causes loss of the vehicle, the life cycle cost to the system is very great. These costs range from cost of the payload to the cost of a possible launch vehicle fleet stand-down until the cause of the failure is understood and the problem fixed. These costs can be reduced by either incorporating some mechanism of fault tolerance, or by simply increasing the probability of correct determination of the location and symptoms of the initial fault. However this failure mode has a small probability, and hence there need to be cost trades to determine the benefit of adding capabilities to the system to increase the fault detection / isolation capability. Lastly, reliability and fault propagation analyses of the system continue, and are used to determine the adequacy of the design, and hence possibly change the design when it is found deficient.

When the system is tested and operated, the actual behavior will most likely differ from the predicted behavior, requiring updates of the various models and analyses performed earlier. This is also the time when various contingency plans are built to determine system and operator response to various failures operationally. These are used to train the operators, as well as continue the search for failure modes within the system.

2.3.5 System Design

The object of any engineering project is to design and build a system, not just analyze it. Ultimately, analyses, requirements, and information are produced to improve and understand the actual design. What are the elements of the SHM design? In the very beginning, these are primarily embedded in the operations, maintenance, and vehicle concepts. If the launch vehicle is assumed to have high availability, but no on-pad maintenance, those concepts will drive the SHM design in certain directions. As the concept is driven to greater degrees of detail, the fault tolerance capability of the system is defined in part by definition of Fault Containment Regions (FCRs) and Error Containment Regions (ECRs). Fault Containment Regions define the boundaries past which faults cannot propagate. This differs from an Error Containment Region insofar as the *symptoms* of a fault, namely an error, can still propagate. As an example, if a ring laser gyro's optical unit fails, it normally will not cause a failure of the electrical components. However, the error, or symptom of the failure, will still propagate into the electronics, ultimately sending bad sensor readings to the nearest processing node. It takes some sort of voting or detection of the bad sensor measurement, with masking or replacement of the bad data with "good data" to create an error containment region. Definition of ECRs and FCRs imply certain types of designs to mask faults (usually involving some sort of redundancy with at least 3 sources of data for the same function) or robust margins so faults in one component do not cause failure of adjacent components.

Later in the process, during the preliminary design phase, the SHM design takes shape in the form of combinations of hardware, software, and operations techniques to detect and respond to failures. The response to failures can be anything from an automatic internal hardware switch from a faulty component to a good component, to a passive sending of data to a human operator, who can then decide what, if anything, should be done given the occurrence of the fault. During later phases, as the design proceeds, fixes to the design and adjustment of parameters occur as the behavior of the actual system is understood. Contingency plans and operations are generated, and are part of the SHM design, since they are the operational responses to faults within the system. Training of operators can become part of the SHM design in order to facilitate the response of humans to fault situations.

2.3.6 Verification and Validation

V&V of a SHM design is a particularly tricky affair, since faults must be injected into the system in order to determine the detection, isolation, and response characteristics of the system. Since the number of failures and situations in which the failures can be injected effectively wind up as an infinite set for a typical aerospace system, some technique of prioritization must be used to determine how much testing is enough. Successful prioritization is based on a complete V&V plan for SHM.

As discussed above, the key to V&V of SHM is fault injection. Thus the first step in developing a V&V plan is to generate a list of the set of faults which are to be injected. This list is generated from FMEAs, other analyses and expert judgement. Thus, informal, early FMEAs must be created to facilitate the SHM design process and the building of capabilities to inject these failures into the system. A failure mode cannot be simulated unless it is identified in the first place. As the design proceeds, the level of FMEA available becomes more detailed, and the planning effort for V&V must reflect these changes as they occur.

Next, the SHM V&V plan must allocate the possible fault types into the various techniques which can be used to determine the response of the system to those faults. Realistically, the V&V of the system will consist of a mix of techniques, such as component, subsystem and system testing, simulation, paper analyses and formal proof (for very high reliability requirements).

As the V&V planning proceeds, requirements upon the system design are levied. For example, if simulation is the technique of choice for a particular fault, then that failure mode must be built into the simulation. The same holds true for testing of the system. The building of software models with simulated failure modes is just as critical as the building of breakout boxes or fault injection devices in hardware. Similarly, formal specification and proof often constrains the design of the device to be "proven". These requirements, both for fault injection and proof, should be levied as soon as possible. This is made difficult since the FMEAs for devices do not typically occur until the design is complete, at which time it is extremely expensive to fix a design problem should one be found. In addition, the constraints of the system design prohibit certain types of faults from being injected, either because of the physical design, or because of cost constraints.

2.3.7 Time Phasing of the SHM Design Process

The initial requirements phase concentrates on generation of a set of requirements and a system concept which are detailed enough for a subsystem designer to begin the fault tolerance and HM design for the subsystem. In order to accomplish this, the first step is to acquire the top level customer demands. Using the IPT approach, the customers and the manufacturers are involved with this process. Quality Function Deployment is one method of acquiring the appropriate requirements. Once this is accomplished, a functional analysis of the system is performed, and a concept or concepts for implementation are defined. There is nothing SHM specific about these tasks. The last part of the initial requirements phase is to determine the requirements for subsystems at a level sufficient for the subsystem designers to determine a first cut at a subsystem implementation.

System Health Management at this point begins to become apparent as an entity within the system. In order to design the system, the basic system implementation philosophies must be specified. From the standpoint of SHM, these requirements include: specification of the fault tolerance level (e.g. Fail-Op/Fail-Safe), the fault classes to be tolerated by the system, the fault classes to be "avoided", (where testing, design margins, and fault free design will be used), subsystem reliability allocations, and other key requirements and constraints, such as cost, availability, and safety. SHM thus plays an important role in the initial requirements development, and must be specified by appropriate requirements just as other functions of the system. The "new" elements introduced into the standard design process are the insistence upon specification of SHM requirements which include quantitative and qualitative features, and up front specification of the fault classes to be tolerated and avoided.

During the conceptual design phase, the focus is to determine basic construction of the subsystems, and to relate these to the system design and interfaces. From a SHM point of view, the initial time to criticality and functional fault analyses must take place to determine subsystem interactions under fault conditions. Behavioral models of the system can aid these analyses. In addition, reliability analyses are continued to determine if the overall reliability of the system is reasonably close to the goal specified earlier. The initial V&V planning also begins during this time.

During the preliminary design phase, the goal is to determine if the design concepts will actually work when the methods of implementation are applied to the concept. From the standpoint of SHM, to determine this, a preliminary FMEA must be performed, which is then compared with the "top-down" functional fault analyses performed earlier. The preliminary FMEA also provides the initial set of faults used in the V&V plan, which in turn provides the fault injection requirements which are used to build failure modes into the hardware models, and to build support equipment hardware with these capabilities. If proof of key algorithms or designs are necessary, these occur during this time phase also. The FMEA provides the basis for the analysis capabilities, including the time domain simulations (based on the models of failure modes within components), and the cost/reliability analyses to determine the system sensor suite.

The detail design phase completes the design, and sets the software thresholds for detection of faults. During this time the test procedures for the fault scenarios are built, and the official proofs are conducted. In addition V&V by analysis and simulation is conducted. Later, during the fabrication, test, and operational phases, these may have to be redone if the system behavior under fault conditions is significantly different from the behavior expected and modeled.

During the fabrication and test phase, the objective is to build the system, and to determine if its performance and behavior are as designed and expected. If they are not, updates to designs, models, and requirements follow. The characterization of the system plays a key role in the final determination of threshold settings. During this time, operational contingency plans are also constructed, and are part of the SHM design

Lastly, during the operational phase, as the system is understood more completely, and for training purposes, fault analysis and contingency plans continue to be worked. Problems found during this phase are fed back into the design process in order to improve the system.

2.4. System Specific Implications

2.4.1. Clean Sheet Versus Retrofit Design

The methodology that this document presents assumes a clean-sheet approach to the system design. Nevertheless, the methodology is certainly valid in applications to existing systems, i.e.: retrofits and/or improvements. But, a number of factors, on the one hand, complicate the process, while on the other, simplify it.

First of all, a great deal is known about the system: subsystem interaction, interfaces, fault "hot spots", environmental characteristics, operational troubles, designs and implementations, to name a few. The amount of known data provides a firm basis for trade studies and new requirements generation. As has been said before, HM functionality should buy itself into the system in terms of the potential lost resources that it protects against. The retrofit and/or improvements will carefully consider this fact, but since the data is available to perform the trade, the justification process is often more straightforward.

Some things complicate the design improvement or retrofit activity. First of all, when making changes to existing systems, the high-level philosophy of dealing with faults, as discussed in Section 2.1, is generally missing or ambiguous at best. For small changes, this is not necessarily a major handicap insofar as making improvements to the system are concerned. However, for large changes, this becomes a greater handicap. Since the system has not been designed with HM considered at the beginning, there are often severe limitations as to what can be achieved. A HM system cannot ultimately fix a bad design. If a simple HM system is desired, then a simple system (from the standpoint of having designed it with HM in mind from the beginning, with steps to minimize failure modes) must be built from the beginning. Often, the HM system can only bandage systems after the fact.

Existing systems are usually hard to change. Since HM functionality by its very nature tends to cross subsystem boundaries, introducing potentially far-reaching rework like this requires the greatest of care.

For large scale changes or improvements, the first step in the retrofit or enhancement activity is to determine the high-level HM requirements, should they be missing or ambiguous (the usual case). This requires a painful analysis of the existing requirements, design and implementation in order to deduce the intended or assumed HM high-level requirements. Section 3.1.3.3.3 discusses the requirements that should exist and be written down at a minimum. Experts on the actual system design exist, and must be brought into this process. In particular, FMEAs exist, and hence there is often little need for "postulating" failure cases in the top down deductive process. Since the particular failure modes are actually known, the typical top down analyses to progressively determine the failure modes of the system are unlikely to be needed. However, it is often the case that the full implications of time-to-criticality, or the timing implications of the failure upon the design, are still unknown, making this analysis useful. The best way to proceed when the data exists is to work "bottom-up", from the existing data regarding failures and the system design. From this bottom-up analysis, once the significant problems are known, and the cost benefit of changes assessed using cost versus system reliability in the trade (as well as operability and other factors as appropriate), the "top-down" process can proceed, designing the changes, and analyzing the system in the manner typically done in the standard process. Thus the procedure for the methodology to be used reverses itself in determining the benefit of making changes, going from a bottom-up process using existing data. Once this is complete, the process reverses itself again to proceed in the standard manner for those portions of the system which are to change.

2.4.2. One of a Kind Versus Production Quantities

The production quantity of a system has a major impact upon the health management system design, and upon the techniques to be used to perform the design. For one of a kind or very low production quantities, building a sophisticated checkout system for the launch system yields less life cycle cost savings than with high production quantity systems. Improvements for the purpose of maintenance and automation of checkout procedures are more important for high production quantities.

For low production quantities, there is a higher risk of design flaw, due to the fact that there are very few systems in existence, with correspondingly fewer opportunities to operate the system. The system must be designed in such a way as to tolerate the inevitable differences between assumptions made in the design versus the actual operation of the system, and to tolerate failures due to design flaws which were not found in testing. For high production quantity systems, the testing of the design is often more thorough, for the high quantities can justify greater amounts of testing and more and higher quality prototypes. In addition since more copies of the system exist, faults and anomalies are more likely to be found, and the results of these findings fed back into the system design, thus reducing the number of flaws inherent in the system.

For systems with one or few copies, there is a lack of quantitative data to historically analyze similar systems (there are usually few of these as well), and one or few systems actually in operation, thus reducing the operating time for gathering of significant data. Even if the data is gathered, if there is only one of the system, and it is operating in space, for example, the updates to the system are limited to software and operations, as opposed to flight hardware. Due to this lack of data, there is a greater reliance on qualitative requirements to drive the design. This is one of the significant differences between space systems and other systems. Space systems, with very few exceptions, are expensive, and limited in number. Due to these factors, reliability data or historical data of any kind is limited in quantity, and in quality as well, for usually each spacecraft is different, thus limiting the usefulness of the data gathered from one system in its application to another. Even for launch vehicles, the historical data is not always useful, for the launch vehicles evolve over time. Within the Titan family of vehicles for example, the current Titan IV bears little resemblance to the Titan I or II. Thus in general for space systems, the historical emphasis on qualitative as opposed to quantitative requirements and measures for design of fault tolerance and health management is quite sensible. So too is the quantitative approach used for commercial systems, whether aircraft or computers, where historical data from operational systems is available in large quantities. One of the major emphases of this methodology is to note that *both* approaches have merit within their respective domains. Space systems will have, and rightly, a greater dependence on qualitative approaches, and commercial or military systems with large production quantities will emphasize quantitative measures and approaches.

—

.

.

—

.

.

—

3.0. SHM Design Process

3.1. System Health Management Initial Requirements Phase

This section of the System Health Management (SHM) Methodology describes the beginning steps of the systems engineering process, emphasizing those elements unique to dependable system design. Health management initial requirements originate from the initial systems engineering activities. Because of close integration between the two, it is difficult to draw a distinct line of separation between general systems engineering activities and health management specific systems engineering activities. The health management system is simply one of many elements which must ultimately be integrated to form a complete system.

The most critical output needed at the completion of the first phase of SHM design is a list of requirements written to a level which allows for the system and subsystem designers to begin assessment of design alternatives at the system and subsystem levels, which accurately define the customer's desires for the system. In order to accomplish this objective, it is necessary to define customer demands in the form of top level system requirements, to lay out an initial system concept, and to generate the next tier of system requirements including SHM requirements. This output is a product of the general system level systems engineering work within which health management plays a major role.

SHM requirements development does not end with the initial requirements definition performed in this phase (see Figure 3.1-1). During preliminary and detail SHM design phase work, requirements and constraints continue to be defined, refined and allocated in an iterative process which develops deeper levels of detail of the system.

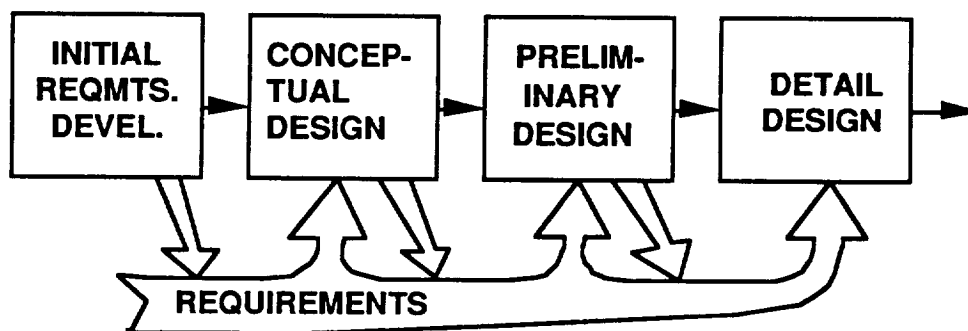


Figure 3.1-1. Initial Requirements Phase is Just the Beginning

3.1.1. SHM Initial Requirements Phase Objective

The objective of this phase is to develop an initial system concept or concepts as a first step toward implementation; to identify, develop, organize, and prioritize initial design requirements relative to SHM, and to identify constraints and figures of merit. The system concept, initial design requirements including fault classes, constraints, and figures of merit are precursors to the SHM conceptual design phase.

3.1.2. SHM Initial Requirements Phase Major Activities

The products of this initial requirements phase are:

- a) a prioritized list of top level customer demands,
- b) a top level system concept or concepts, usually presented in the form of system block diagrams and mission and operational timelines and including top level operational and maintenance concepts,
- c) a set of SHM requirements usable by system and subsystem designers to perform subsystem level design trade studies, and to define the major elements and implementation concepts for the subsystems,
- d) a set of fault classes for the hardware element, the software element, and the operations element of the design,
- e) and a list of constraints and figures of merit.

As shown in **Figure 3.1-2**, the SHM initial requirements phase consists of three major activities: top level customer demands, system concept and SHM requirements. The first two activities are major systems engineering activities within which SHM activities are integrated.

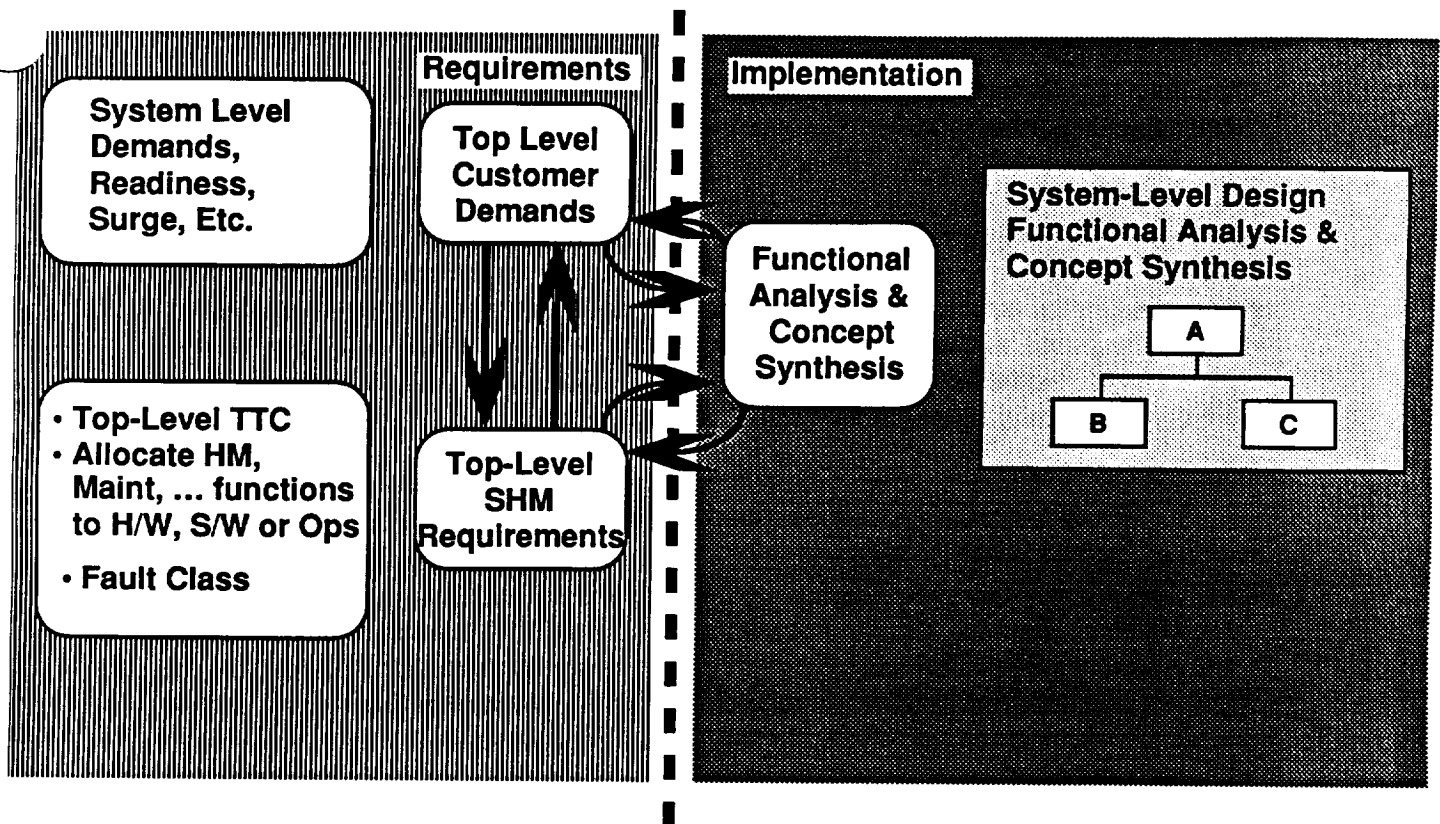


Figure 3.1-2. The System Health Management Initial Requirements Phase

The first activity in development of any system is to define the requirements for the system. Typically, in the past, requirements were assumed to be “givens.” The customer stated the requirements, and the contractor implemented them. For development of launch systems such as NLS, the typical customer has been the government procuring agency. Today, however, it has become clear that in order to generate correct requirements, it is necessary to access the knowledge of the users and builders of the system as well. The definition of the customer must be expanded to include any group who is affected by or affects the system being designed. As will be discussed below, there exist methodical processes for capturing knowledge and desires from a diverse group of individuals. Although it is “motherhood” to say that it is essential to determine the correct requirements, it is very seldom actually accomplished to the degree of completeness and accuracy needed. This has in the past led at a minimum to large cost and schedule overruns for programs as they find problems later in the design process based on incorrect or incomplete requirements. System failure has also been a consequence of incorrect requirements. Requirements development is now recognized as a major phase in the system design process.

The second major activity of this phase is to create a concept or concepts for the system from the top level customer demands (inputs and requirements). It is often true that this activity takes place in parallel with the system requirements development activity above, in the sense that the users, operators, builders, and procuring agency usually have some idea of the type of system or implementation that is appropriate for the specified requirements. The top level system concept development activity involves a set of trade studies at the highest level, to determine top level system implementation options. These trades usually involve the critical program technical issues, whatever they may be, and a definition of the major system elements required to perform the mission, along with an initial definition of the mission itself. Initial operational and maintenance concepts are also developed. This activity again ideally involves our broadly defined customer.

The third major activity that occurs in this phase is to derive from the top level system concept (also operational and maintenance concepts) the next level of requirements which define the program implementation philosophy. For example, at the system level, the major functions which must be performed are defined in the system requirements. However, there is not yet a clear statement of how these functions are to be implemented. The subsystem designer needs to have clear definition of applicable system level requirements, including SHM, in order to begin work on the design. As an example, for NLS, the top level of requirements for the system may require the system to be 99% reliable for the flight phase, 95% available for launch on a given day, and to not jeopardize human safety. The system concept will define the system to be a launch vehicle of a particular size and performance, with a certain type of operational concept, and to have a set of functions it will perform, such as guidance and control, power, telemetry, etc. The next level of requirements necessary for the subsystem designer of SHM is to specify the reliability targets for the subsystem, specify the level of fault tolerance (one or two fault tolerance, for example), specify the fault isolation capability (e.g. diagnose failures to the LRU level), and so on. This is the level of requirement which a designer can work with to begin the subsystem design.

3.1.3. SHM Initial Requirements Development Approach

The three steps of our System Health Management initial requirements approach are shown in Figure 3.1-2. They are described in detail below.

3.1.3.1. System Requirements = Top Level Customer Demands

The first step in the development of the system is to capture and encapsulate customer demands in a form usable by all parties involved in the development, manufacture, and operation of the system.

3.1.3.1.1. Integrated Product Development

The essence of developing correct requirements is to get all of the parties who have an interest in the system to communicate their needs and wants, and then to develop a consensus as to the importance of these various desires (Figure 3.1-3). A team which includes all of the concerned organizations in this manner is referred to as an Integrated Product Team (IPT).

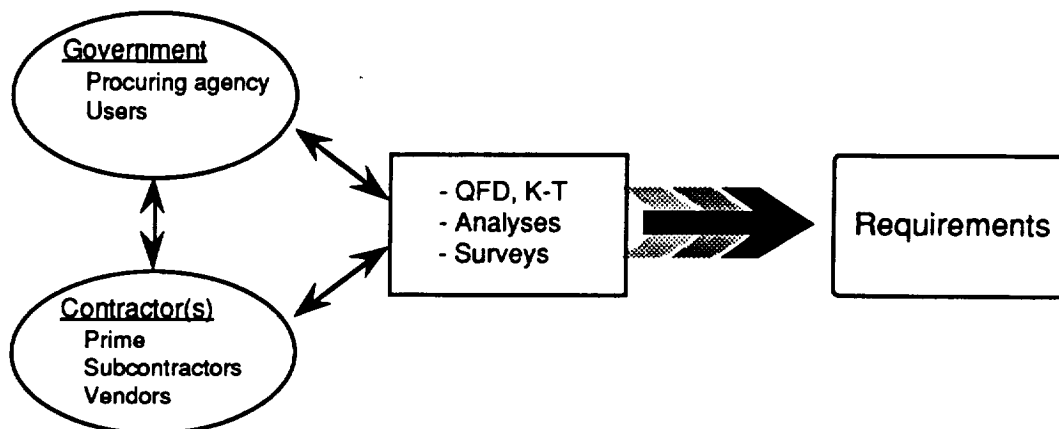


Figure 3.1-3. Joint Effort Requirements Development

There are a number of mechanisms or tools to achieve this consensus within an IPT. One of these, which will be described in more detail, is Quality Function Deployment (QFD). Ideally, the complete intent of the customer is captured within the system requirements. However, this ideal is rarely achieved. In order to better achieve this objective, it is crucial that the procuring agency, user and contractor iterate, negotiate, and develop the requirements using shared analyses and negotiations. QFD is one mechanism for just such a development. This document shall assume use of the QFD process as a typical Total Quality Management (TQM) process for systems engineering.

The QFD process is only one way of establishing a set of initial requirements. There are others which can be used, such as Kepner-Tregoe (described in more detail later). The important point to note is that communication between all concerned parties must take place, and the requirements must ultimately reflect the concerns and needs of these organizations and individuals. This can only be accomplished when the procuring agency, manufacturers, users, and operators are part of the process, and their inputs accounted for.

3.1.3.1.2. Quality Function Deployment

QFD is a procedure and process for development of requirements for a system or product (see Figure 3.1-4). It is essentially a mechanism for ensuring that all of the parties involved with the system have their concerns reflected in the requirements for a system, and then deploying these requirements in a consistent manner throughout the system as it evolves. This section shall discuss the initial portion of the QFD process, through the development of the requirements “tree”, and the Level of Importance scoring for these requirements.

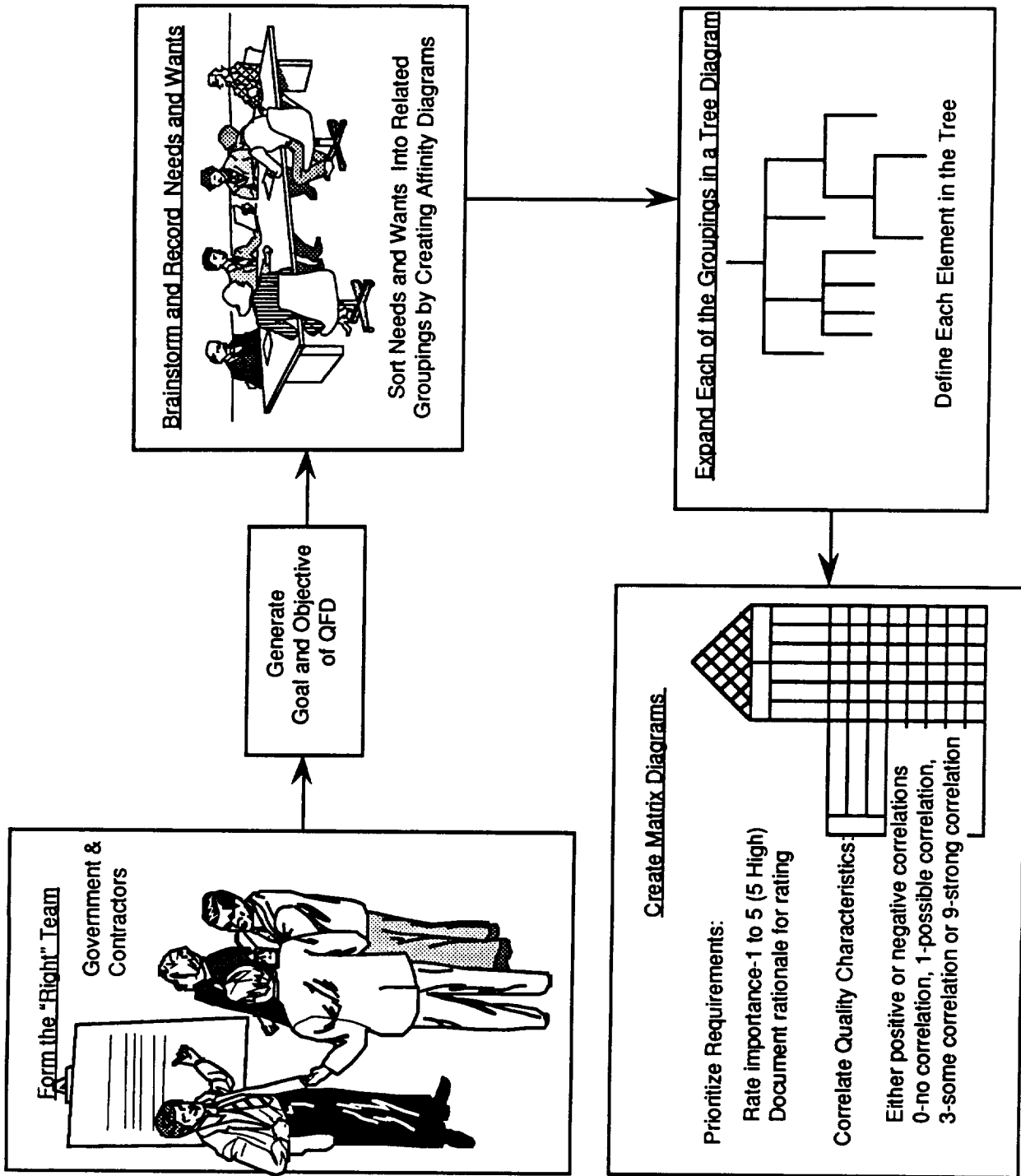


Figure 3.1-4. QFD Overview

Essential to the process is to have all of the appropriate organizations and key personnel present. In the case of a system such as NLS, the procuring agencies, the payload community, the manufacturers (both prime contractors and subcontractors), the operators at the launch facilities, and mission experts must be participants in the requirements generation.

As an example of a program which has used QFD during this phase, the Advanced Launch System (ALS) program undertook a QFD some time after the government had levied its initial requirements using standard procedures. The former ALS Joint Program Office Director indicated that Quality Function Deployment (QFD) sessions between the government and contractors substantially improved the expression of the government intent in the ALS Systems Requirements Document. Had QFD been used earlier in the ALS program, even better requirements would have resulted.

3.1.3.1.3. Goals, Products, Constraints

The first step is to determine the goal and product of the QFD, any constraints and groundrules which are to be assumed, and the customer for the product for which the QFD is being done. For NLS, such a goal could be something like, "determine the requirements for a family of new launch vehicles to provide assured access to space." The product could be "a set of matrices defining the requirements and prioritization of these requirements for the launch system." There could also be lower level matrices which define and prioritize what are referred to in QFD language as the "quality characteristics." Constraints are items such as, "assume that certain manufacturing capabilities will be used" (other examples are shown in Figure 3.1-5). In other words, if certain groundrules are to be used based upon political, funding, or other factors, these need to be specified so the entire team is aware of them and can account for them appropriately later in the process when changes occur. The customers in the QFD are those individuals, groups, or organizations who will use the system. For a launch system, this includes the procuring agency, the payload users, and the system operators.

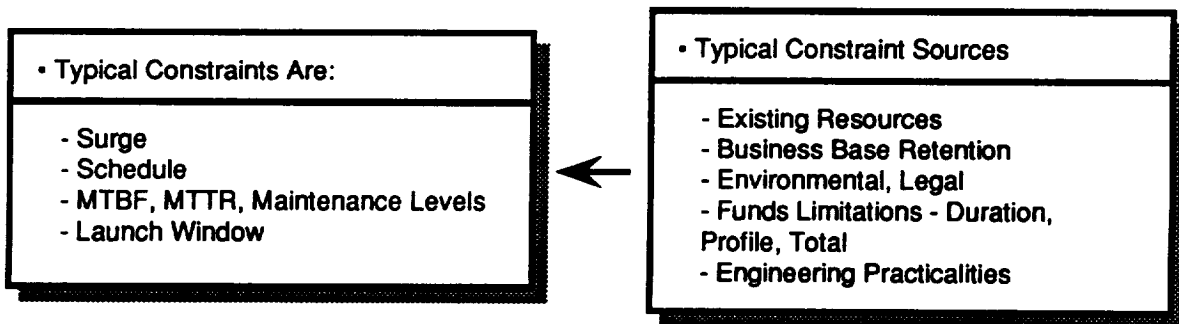


Figure 3.1-5. Constraints

3.1.3.1.4. Requirements Generation and Prioritization

Once the goal and constraints have been specified by the team, the team uses brainstorming techniques to come up with a list of the needs and wants which the system must satisfy (see Figure 3.1-4). At this stage, these are in no particular order, and can be at various levels. In other words, some of the needs will be high level items such as “assure safety to personnel and facilities”, whereas others could be lower level items such as “the system shall have a mean time to repair of less than 6 hours.” This somewhat random collection of items is then organized by the team into several groups of related items using a technique called “affinity diagrams.” For example, all items related to safety would wind up in one group, all items related to system performance in another, and so on. These groupings can then be organized into a tree structure and names given for each group. The tree structure is then filled out for any missing elements, and is driven down to a level which the team feels is satisfactory for specifying the requirements to be met by the system. Each element in the tree is given a group approved definition.

The next step is to prioritize the requirements. To do this, each requirement is given a ranking from 1 to 5 in importance, and the rationale for this rating is documented. It is important to note that the customer(s) gives the ranking, not the system manufacturers. The importance of requirements is driven primarily by customer needs and desires, not the manufacturer’s capability to meet those needs.

The result of these efforts is the production of a set of well defined requirements with attached rationale, definitions, and levels of importance. These are the top level system requirements, which should be documented as such. Usually, system requirements are not given levels of importance. Giving them such a ranking is an important improvement over the standard procedure, for it is in fact true that requirements are not equal, and the designers and builders of the system need to be aware of this prioritization.

The major objective of this process is to capture and clarify customer demands in a manner which is clear and precise to those who must ultimately fulfill those requirements. The reason manufacturers are part of the process is that they can often help the customer define what the true needs really are, and also can temper these needs with the reality of what can be actually built. The requirements and system which ultimately result are a compromise between the ideal needs of the customer and the actual capability to meet these needs.

3.1.3.2. System Concept Development

The development of the system requirements and customer demands is followed by development of the system concept. At this time, the emphasis switches from the customer community to the manufacturers and vendors. This does not mean there is no more involvement by the customer. On the contrary, there is continuing active involvement by the customer all throughout the design process. In the first phase above, the customer involves the designers in order to temper the requirements and needs with the reality of what is achievable. In this step, where the first look at implementation begins, the designers need to involve the customer to aid with interpretation of the customer requirements, for it is usually true that the implications of a given requirement are not fully understood until the system(s) which results begins to appear. As the candidate implementations begin to develop, it is quite possible that the requirements will to be changed to reflect the impossibility of achieving certain goals, or conversely, to account for the ease of achieving requirements which were thought too difficult or less important for various reasons. In addition, they change as understanding of the system allows for clarification of the requirements.

The essence of system concept development is first, the generation of a large set of ideas or concepts which are candidates to solve the problem at hand. The second step is to winnow these concepts down to a single baseline concept if possible, or to a small number of promising concepts if not. In order to accomplish these tasks, a systematic process must provide the means to generate the appropriate figures of merit to judge between implementation options, and also generate information which can help identify trade studies which affect the various concepts. These trade studies also use the figures of merit discussed below.

Lastly, it is often found that there is insufficient information to make judgments concerning certain options. In these cases, further analysis or information gathering is necessary, or it may be necessary to keep several system concepts open until later in the design process until the required data is available.

3.1.3.2.1. Figures of Merit Development

In order to evaluate system concepts, the figures of merit (FOM) for the system must be developed (see Figure 3.1-6). Figures of merit are quantifiable evaluation criteria used to judge one design concept against another. From a Kepner-Tregoe perspective, figures of merit can be thought of as the weighted "wants" part of the Kepner-Tregoe decision analysis. The figures of merit should be sufficiently developed to allow fair and equitable grading of competing design concepts. Figures of merit should have clear definitions, be practical (physically meaningful), and quantifiable (as easy to compute as possible) for the designer to use in trade studies.

Figures of Merit Are:

- Quantifiable For Trade Studies
- Relatable to System or Subsystem For Guiding Design Process
- Often Related to Cost
 - Life Cycle
 - Recurring
 - Non- Recurring

Figure 3.1-6. Figures of Merit

Decisions in early conceptual design are often more difficult to make than later in the design process. This is because metrics and detailed design information often do not yet exist. It is recognized that application of figures of merit is often more difficult for the top level systems decisions because of scarcity of data or limited time and resources to accurately model complex systems. The effort spent in developing figures of merit is worthwhile, however, to enable more intelligent decision making for systems concepts. Reference [JORD 91] details weighted evaluation criteria developed by the NLS Evaluation Criteria working group that provide a framework for launch vehicle figures of merit development. Within the QFD process, the requirements with their quantitative level of importance rankings become the most obvious figures of merit. The way they are used is to first develop the "A-1" matrix, which compares "whats", the requirements, with "hows", or the characteristics of the implementation (see Figure 3.1-7).

The "hows" are termed "quality characteristics". For a launch vehicle, a typical "how" might be "guidance accuracy", which is a controllable characteristic of the design. This quality characteristic is correlated with each requirement one at a time, determining what correlation there is between changing the guidance accuracy with a particular requirement, such as 'provide a smooth ride for the payload'. The correlation is indicated with a number, which in QFD is customarily chosen as a 0, 1, 3, or 9, which range from "no correlation", "possible correlation", "some correlation", and "strong correlation". This number is multiplied with the level of importance for that requirement, and the resulting number is put in the matrix. Once this is done for each requirement, the correlation figures are summed for each quality characteristic. If a particular quality characteristic correlates strongly with many important requirements, then it will get a high ranking, and thus the importance of that characteristic will be emphasized in the design in order to best meet customer demands.

<div> <div>Correlation Code</div> <div> 0 None 1 Weak 3 Moderate 9 Strong </div> </div> <div> <div>Customer Needs "Whats"</div> <div>Implementation Methods - "Hows"</div> </div>		Valid./Cert. of Qual. Char'tics	Normal In/Egress Capability	Emergency Egress Capability	Life Support Conditioning	Crew/Cab Environment	Ext. Env. Tolerance Factors	Safety of Destruct System	Veh. Tolerance to Induced Env.s	Non-Abort Insertion	Ability to Abort Successfully	System Status Determination	Rate of Importance	Sales Point	Status Now	Plan to Get to	Importance Ratio	Absolute Weight	Demand Weight	Relative Ranking
Human-Rated Launch System Capability	Don't Impair the Crew	0	1	1	3	9	1	1	1	3	1	1	5	1.2	4	5	1.25	7.50	12	2
	Don't Impair the Cab	0	1	0	1	3	9	9	3	3	1	1	4	1	3	5	1.67	6.67	11	2
	Provide Crew Comfort	0	1	1	9	9	1	0	1	1	1	1	2	1	4	4	1	2.00	3	3
	Crew Safety Awareness	1	0	1	0	1	0	1	0	1	1	9	2	1	2	3	1.5	3.00	5	3
	Gr Awareness of Crew Safety	1	1	1	0	0	1	1	0	1	1	9	1	1	3	3	1	1.00	2	3
	Non-Flight Ops W/ASR	0	9	9	3	1	1	9	0	0	1	9	5	1	3	5	1.67	8.33	13	2
	Flight Ops W/ASR	0	0	1	3	9	0	9	9	9	9	9	5	1.2	3	5	1.67	10.0	16	2
	Crew Safety Robustness	0	1	3	1	1	0	3	9	9	9	3	4	1.5	1	3	3	18.0	29	1
	High Degree of Prod Assurance	9	0	0	0	1	1	1	1	1	1	1	3	1	2	4	2	6.00	10	2
	Weighted Percentage	2	4	6	5	9	14	12	12	12	12	11								
Relative Ranking		4	4	4	4	3	1	2	2	2	2	2								

Figure 3.1-7. A-1 Matrix Example

It is possible at this point to use the rated quality characteristics (the ranked quality characteristics as discussed in the previous paragraph) as figures of merit for design options. The rated quality characteristics provide a quantitative measure of which controllable features of the system best meet the customer demands and requirements. It is a relatively straightforward exercise to now compare design options using their correlation to quality characteristics. For example, if a specific quality characteristic which is highly rated is 'launch vehicle flight reliability', it is now possible to compare several design options against this particular quantitative measure. The goal in QFD is to create quantitative measures against which designs can be assessed. Thus, if designs are to be compared, these quantifiable attributes are the figures of merit which will be used to make the comparison.

Other figures of merit are frequently used, and are of value. Implicitly or explicitly, cost is a major factor, and is used as a FOM. Technical and schedule risk are also major figures of merit. These are typically embedded in the A-1 matrix itself, and hence are a standard part of the QFD process. Performance factors are also necessary. It is important to note that figures of merit are used for both relative evaluations of different designs, and also for absolute 'gates' which must be met for the system to meet the given customer demands. As an example, reliability can be used as both an absolute criterion which must be met as a minimum. Also, should the system concept be judged adequate in this regard, it can be used as a relative assessment factor to compare differing concepts or designs. Figures of merit are only of value insofar as they are used for the primary goal, which is to judge design concepts and options.

3.1.3.2.2. Definition of Trade Studies

Another necessary component of the system concept definition phase is the identification and use of trade studies. Identification of appropriate trade studies, and their relative priority, is an important task.

Each quality characteristic defined in the QFD process can, and usually does have an associated set of trade studies which can easily be identified by the experts for that area. These trades usually assess design alternatives within that particular area.

Another mechanism which can be used to identify trade studies is to create an "A-3 matrix" (shown in **Figure 3.1-8**), which compares each quality characteristic to all other quality characteristics. This is often pictured as the "roof" on the "house of quality". This comparison looks for positive or negative correlation's between various controllable aspects of the design. For example, is "reliability" correlated with "guidance accuracy"? As an example, if the reliability of the system is improved or decreased, will guidance accuracy be affected positively, negatively, or not at all? Correlations such as these point out (in particular for negative correlation's) the need for a trade study to determine how much of one characteristic can be compromised to achieve another characteristic. These trades cover the cross specialization trade studies which must be performed to optimize the system as a whole. At this stage of the system definition, these are probably the most important items to identify and assess.

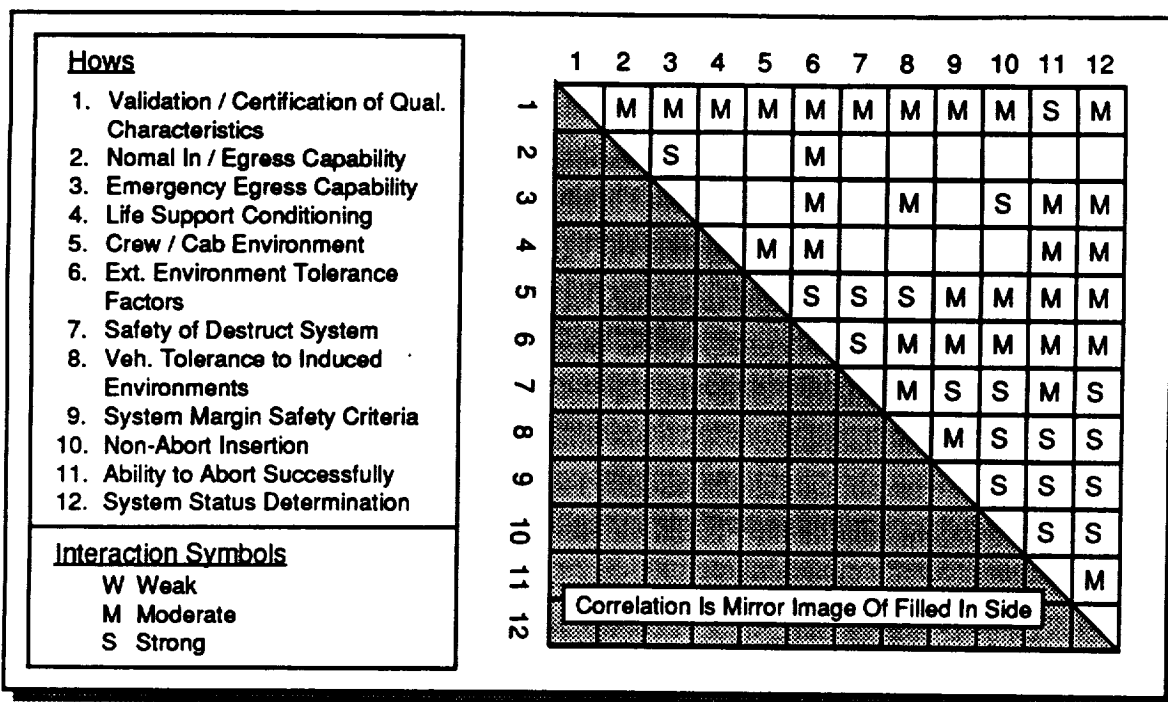


Figure 3.1-8. A-3 Matrix

3.1.3.2.3. Concept Evaluation

The goal of the foregoing efforts is to generate the information necessary to assess various design options for the system, whatever they may be. They are simply tools to help the engineer or designer communicate with other specialty areas, and to help identify where problem areas are likely to be. Ultimately, though, the decisions come down to the team of designers working the system concepts themselves.

As noted above, the QFD process for design assessment provides the weighted quality characteristics as the primary figures of merit, and correlates these to the specific design options which the engineers define.

A shortened process has also been used: to compare designs immediately using only the requirements with their levels of importance. This approach attempts to bypass the step of generating the quality characteristics, and has been successfully used. However, it has been noted that this shortcut can lead to some confusion, for it is not always clear how a particular design relates to a specific requirement. The usual method, which is to generate the controllable characteristics of the design before doing trade studies, allows for a much simpler comparison of designs versus quantifiable design characteristics.

Another method is the Kepner-Tregoe (K-T) process. In the K-T process, identification of design “wants” and assignment of weighting factors to the wants is a commonly used method of figures of merit identification and weighting. Constraints can be viewed as the “musts” in the Kepner-Tregoe process. Constraints arise from many domains, not only from physical laws and the state of the art of engineering technology, but economic, programmatic, political, and national resource domains which can be as equally inviolate as engineering constraints.

Ultimately, the objective of this activity is to create a system concept which meets the objectives of the top level customer requirements. The QFD or K-T processes, or any other process, are simply means to help stimulate thought and document the information which is generated by those persons performing the work.

During this phase of a system’s development, it is often found that there is insufficient information to be able to decide among various candidate implementations. There are often situations, in particular where new technology is involved or substantial complexities which are not understood, where the knowledge to make responsible decisions simply is not there. In this situation, a quick probing of lower level detail is necessary, at least in a narrow piece of the overall system. Rapid prototyping techniques are a means to alleviate this problem.

Rapid prototyping is essentially a way of saying “quickly develop a design and implementation”. In rapid prototyping, it is less important to determine all of the ramifications of the design than to quickly drive the design down to a low level, with the objective of rapidly flushing out the major issues involved with the new concept, technology, or implementation. At the requirements and system concept phase, it is particularly important, for there may be low level complications which can be unsolvable, or would require major adjustments to the system concept. It is even possible that the customer demands may have to be modified, if there is no other way to achieve a particular customer goal. Thus it is essential when there are major uncertainties to balance the methodical top down approach with quick engineering problem solving and implementation.

Another approach to dealing with the inability to choose among implementation options is simply to wait until later in the process to make the decision, to carry more than one concept until later in the process. This is an acceptable, but perhaps more expensive option. For the purposes of this document, it shall be assumed that there is a single baseline concept, even though practically there may be several. In that case, the following processes would potentially be replicated for the number of concepts which are carried to later stages.

3.1.3.2.4. Integrating SHM With Operations and Maintenance Concept

The health management system is closely integrated with the vehicle operations and maintenance concepts and the associated data management plan. Requirements for data management related to SHM should be addressed in the data management plan as shown in Table 3.1-1.

SHM Data Handling Requirements Should Address:

- **What Information Is Generated**
 - **How Data Is Transformed Into Information**
- **Why and When Information Is Utilized**
 - **Database and Retrievability Characteristics**
 - **Data Compression/Decompression Needs**
- **How and By Whom the Information Is Communicated and Used**
- **Where the Information Is Recorded, Stored, and Displayed**
- **What Equipment Routines and Personnel Procedures Are Applied**

Table 3.1-1 Health Management Data Handling Plan

Shown in Figure 3.1-9 are some of the elements of the health management system that integrate with operations and maintenance databases. Opportunities abound for cost savings by efficiently integrating the health management system with the operations and maintenance plan. Common mechanical and electrical interfaces, avionics modules, and general software jointly developed by operations, maintenance, and SHM cooperation can contribute major cost efficiencies to launch systems programs.

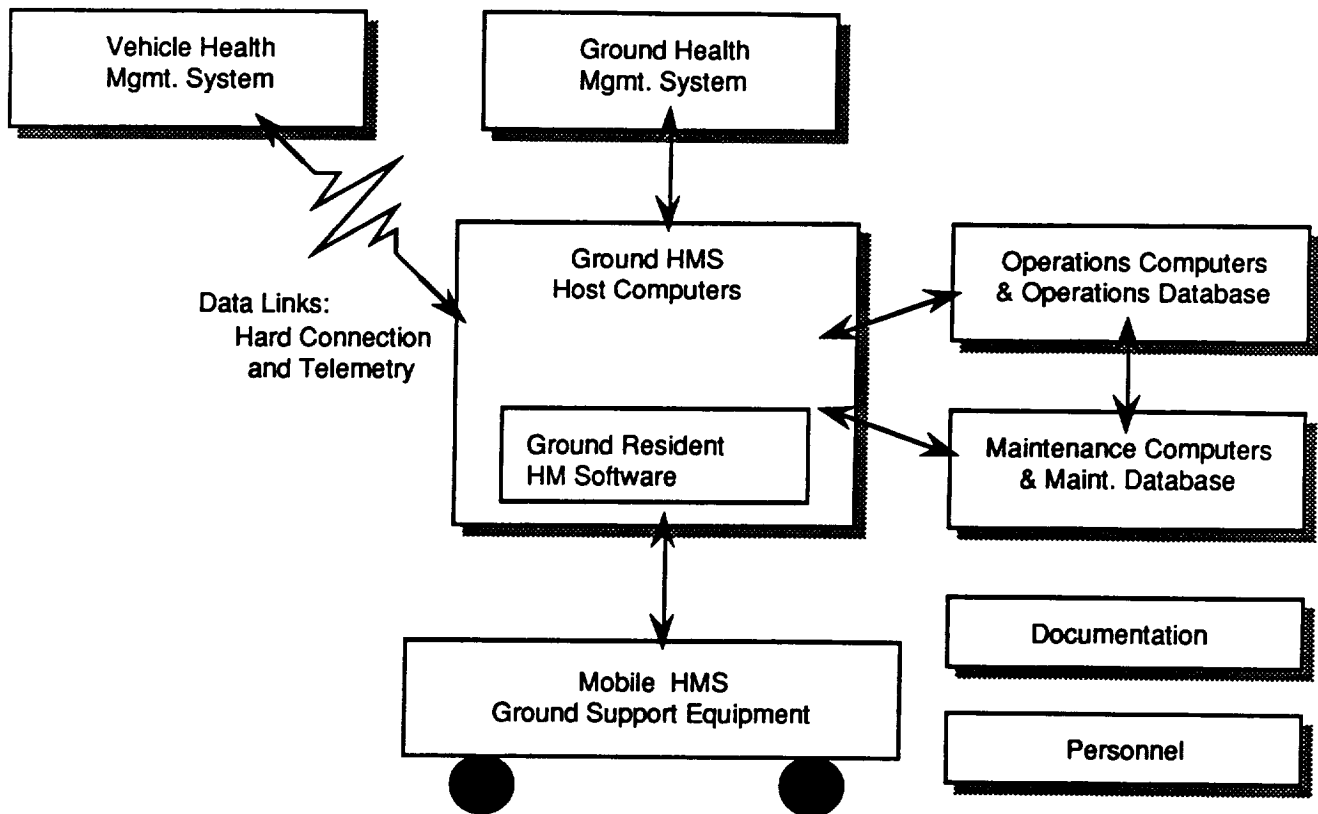


Figure 3.1-9. Integrating HMS With Operations, Maintenance and Other Ground Systems

3.1.3.3. SHM Initial Requirements Development

The emphasis so far has been on system level issues. It is at this point in the design process that we will begin to focus on SHM specific issues, starting with generation of requirements specific to the HM design of the system. The goal of this phase of the design activity is to generate a set of requirements germane to a particular subsystem which are sufficient to allow the engineer to come up with an initial concept and design. To do this, target allocations of cost, weight, volume, power, reliability and others must be specified to a level sufficient for the subsystem engineer to determine a concept which is implementable in hardware, software, and operations. These subsystem requirements can only be generated once there is a system concept in place.

3.1.3.3.1. Requirements Versus Design Guidelines

The system requirements phase is complicated by the breadth of health management, the numerous and sometimes complex interrelationships of parameters associated with SHM, and the still only partial understanding of how to capture the various aspects of SHM within requirements statements. Additionally, requirements must be differentiated from design guidelines. A requirement is a verifiable statement of system performance allocable to something in the design. Statements for which no means of verification can be established are design goals and not requirements. Well defined requirements should define "what" is to be done (function), "how well" the function is to be performed (performance requirement), and "how" the requirement is to be verified. Requirement excellence occurs where requirements are complete, correct, understandable, and unambiguous.

A design guideline, on the other hand, is a statement of a process or goal to be considered in the process of the design. An example could be "the SHM design shall be as simple as possible". Guidelines such as these cannot be verified, although they are in fact good design practice. The key point is to ensure that if they are stated, that they are stated as guidelines, and not as requirements, for requirements must be verifiable.

3.1.3.3.2. SHM Requirements

As discussed in Section 2.1, the methodology requires specification of *both* quantitative and qualitative requirements (see Figure 3.1-10).

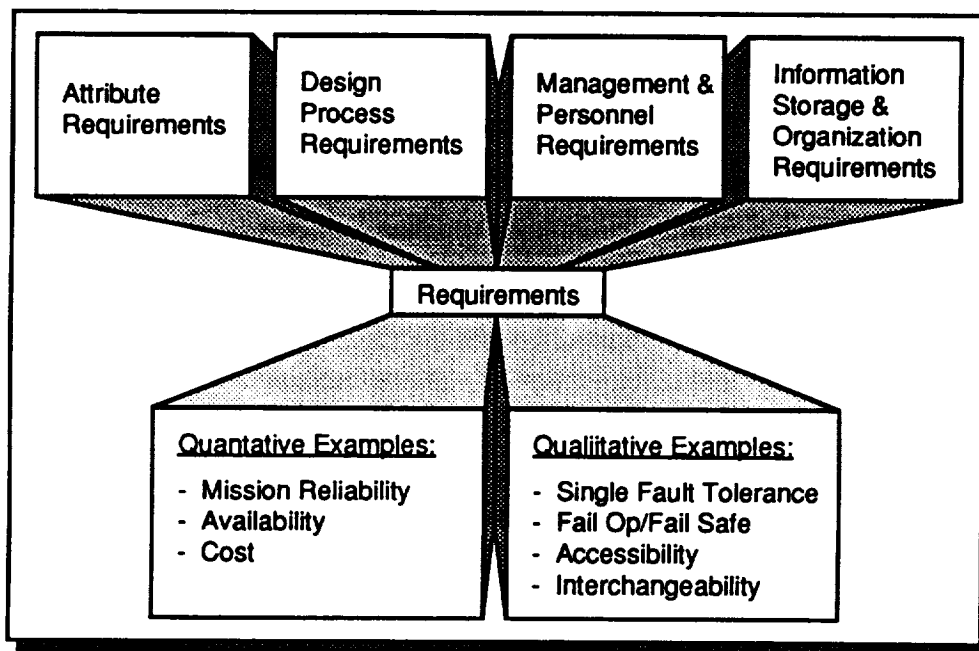


Figure 3.1-10. Qualitative & Quantitative Requirements

Both qualitative and quantitative requirements are used to place absolute constraints on the system, to ensure the correct technologies are incorporated to achieve the specified performance. However, since any particular estimate of the performance (such as reliability or maintainability) is debatable, the actual validation of the system cannot be based on quantitative estimates, but rather on testing the system based on a qualitative requirements. Thus qualitative requirements are necessary to allow the generation of the V&V requirements. Quantitative measures are also used to specify fault avoidance design margins and to assess the relative merits of different design options.

Requirements must be specified to a low enough level to unambiguously define the customer desires. This includes at a minimum the specification of the set of faults which the system is to protect against, as well as the level of that protection (as specified by quantitative and qualitative requirements). Process control requirements are often the means used to assure system performance where quantitative requirements and an associated method of validating them has not been successfully devised. Many government and industry standards are combinations of elements of quantitative, qualitative, process, personnel, and documentation requirements.

Shown in Figure 3.1-11 is a "generic" dependable system attributes tree developed within Martin Marietta using a multi-disciplinary team. Dependable system attributes are closely related to the specific design features of the Health Management System, and hence, initial requirements for the SHM system will address many of the attributes shown in the tree. Moving from the generic to a National Launch System specific, the operability branch of the system robustness tree developed by NLS, and shown in Figure 3.1-12 identifies many of the same elements which exist in the generic dependability tree, but puts them in a slightly different structure. This is typical of a QFD requirements development process. It is less important for the two "trees" to match exactly than that the significant requirements appear in the trees somewhere.

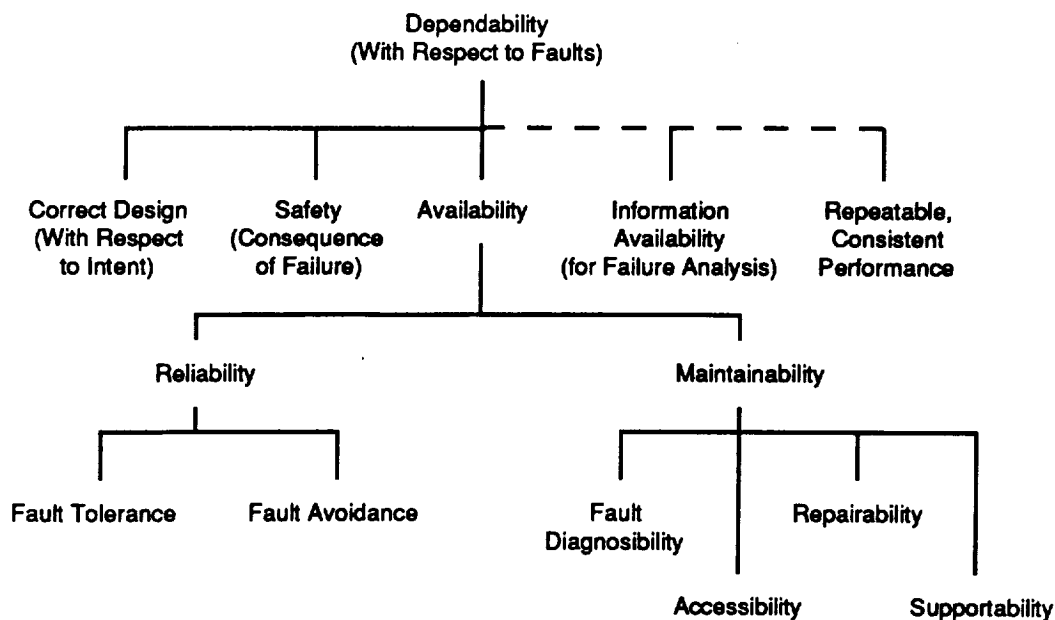


Figure 3.1-11. Dependable System Attribute Tree

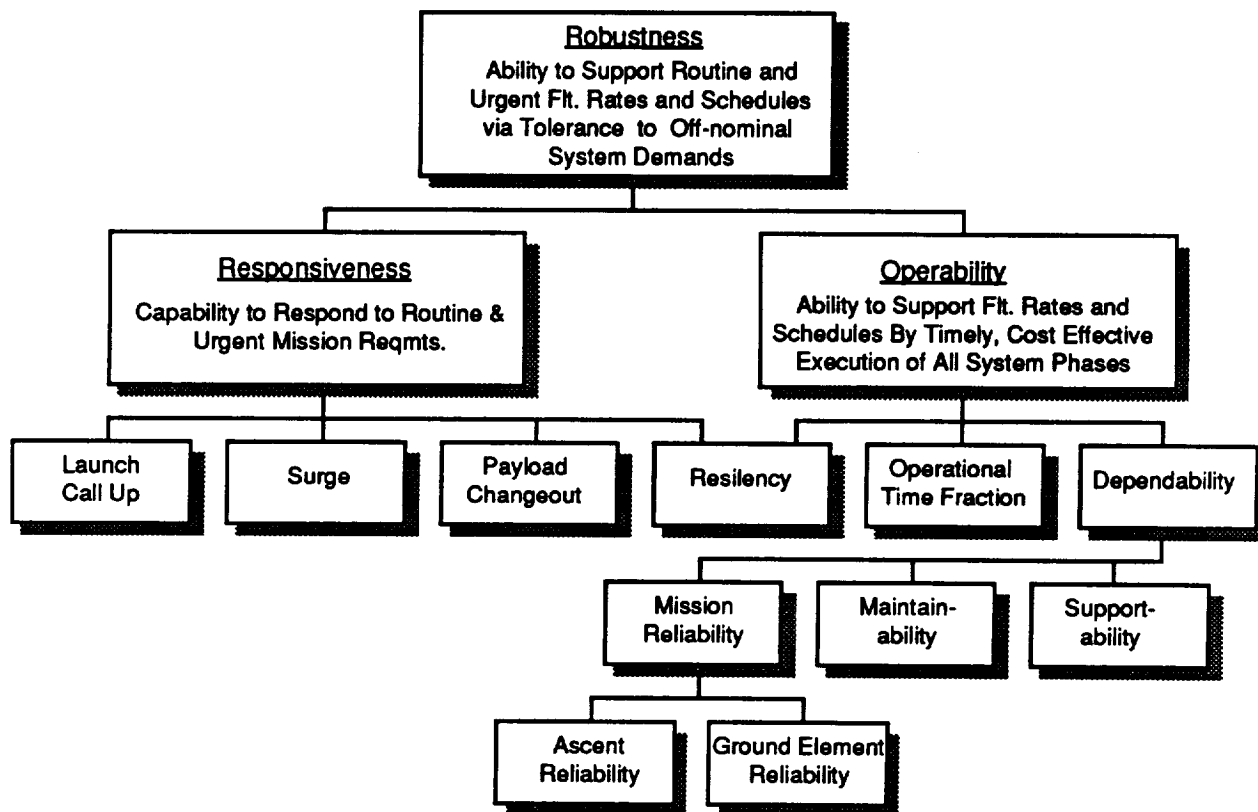


Figure 3.1-12. NLS Robustness Tree

From a SHM perspective, the expression of SHM requirements in terms of quantitative parameters related to the attributes tree is highly desirable. The engineering team associated with SHM initial requirements development should explicitly identify quantitative parameters for the tree, and attempt to establish quantitative requirements values for the parameters as much as possible in the initial requirements phase. Better quantitative values will emerge as the health management system design proceeds the preliminary and detail design phases.

As shown in the dependable system attribute tree, reliability can be attained by the proper combination of fault tolerance and fault avoidance. The initial requirements should allow the designer freedom to choose the right mix of fault tolerance and fault avoidance. Each vehicle subsystem is likely to have a different mix of these attributes. As examples, avionics systems tend to use fault tolerance, whereas mechanical systems place greater emphasis on fault avoidance.

Because requirements impact both the HMS design and other aspects of the system design, the requirements development process continues to benefit from the systems level interaction and communication of government and contractor personnel as shown in Figure 3.1-3.

Clearly the elements of the dependability attributes tree are related to the HMS. However, it is also true that other items, such as cost and risk, also affect the HMS design. The situation is really not so different from other subsystems. These systems also are affected by cost and risk constraints, and potentially a number of others. This document will not attempt to address all possible requirements which can affect an HMS design, for this varies with each possible system. However, there will be discussion of different types of requirements as appropriate to discuss and clarify the SHM Methodology. In the next few sections, the major requirements which affect SHM will be discussed.

3.1.3.3. Development of Fault Tolerance Requirements

The SHM initial requirements development is not complete without establishing the fault class requirements for the hardware, software, and operations elements of the design. Often, requirements such as “no single point failure” are insufficient to clearly delineate the fault tolerance properties required of the system. Specifically, to what faults does the requirement pertain to? It is usually assumed to apply to permanent hardware faults only. This is a very limiting assumption. Nor does it always capture the intent of the customer. For example, are multiple transient faults included, or latent faults? In the case of latent faults, a difficult scenario is as follows. Assume a latent fault exists in the system, but does not appear until another fault causes a switch to the location of the latent fault. Now the system has to deal with the equivalent of two near-simultaneous faults. Is this covered by the “no single point of failure” requirement? Other examples abound. What about a software bug in the flight data system which can cause loss of the vehicle, or a bad command?

Without specification of the faults which the fault tolerance requirements are intended for, it is open to interpretation, with potentially very different designs and fault tolerance capabilities built into the system. By ignoring or excessively limiting the fault set the system is to tolerate, system dependability is immediately compromised relative to these ignored or overlooked faults. Too frequently, detailed Failure Mode and Effects Analyses (FMEAs) have been conducted relative to a woefully deficient fault set, greatly overestimating the actual system reliability and fault tolerance.

A sample fault class list is given in Table 3.1-2.

<ul style="list-style-type: none"> • Root Causes <ul style="list-style-type: none"> - Design - Specification - Part Lifetime - Physical Performance - Mental Performance - Malicious Intent - Environmental • Temporal Manifestation <ul style="list-style-type: none"> - Permanent - Transient - Latent - Intermittent 	<ul style="list-style-type: none"> • Physical/Logical Location <ul style="list-style-type: none"> - Electronic - Structural - Software - Mechanisms - Engines - ... • Functional <ul style="list-style-type: none"> - Propulsion - Power - Guidance & Control - Data Management - ...
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Table 3.1-2. Fault Classes

A fault class is simply a logical grouping of faults, which allows a general discussion of the much larger set of individual faults (the fault set). Use of classifications such as those in the table allow general requirements for fault tolerance to be written without having to enumerate the total list of individual faults. There are many possible classifications of faults, only some of which appear in the table. This particular set of classifications are those which have been found to be useful at this point in the SHM design process. It is essential to consider the means or processes by which faults occur. If only permanent, random part failures are of concern (which should be apparent by the tone of the whole document to be a ridiculously limiting assumption), then that should be stated explicitly, and the other root causes and temporal manifestations explicitly stated as to be assumed to be "designed out" or "avoided." As should be clear by now, other faults should be considered. When considering what is of most use to the subsystem designer, it is clear that, at a minimum, the root causes and temporal manifestations of the faults need to be considered. A system to handle many transient faults (not necessarily simultaneous) will have different designs and characteristics than those which must handle only permanents. Designs to deal with latent faults must either effectively add one fault tolerance level to the design (from "single" to "double" faults), or have extensive built in test capabilities, particularly for components which are normally powered off. Also, there are often particular limitations on these requirements. For example, it is common that "single fault tolerance" applies only to the electronic components of the system, not mechanical or structural components. As will be shown by example later, this is not always true. Some faults are so significant for the system that they are handled as a special case, such as engines for a lower stage booster. Since "engine-out" capability is so critical to the performance of the system as a whole, and modifies the entire system design, it must be specified at the very beginning of a launch vehicle design one way or the other.

After a potential list of fault classes is defined for the system, the next step is to use this list to write the fault tolerance requirements. It is recommended that at a minimum the fault tolerance requirements during this phase of the design be specified at least for the "Temporal Manifestation" and "Root Cause" classifications at the system level (see Table 3.1.2). However, in addition, it is also often necessary to include particular faults related to the functions and/or implementations. For example, for a launch vehicle, due to the importance of engine performance to the overall system design, determination of whether "engine-out" capability is to be incorporated is essential. Engine-out is in fact a fault tolerance specification for the system as a whole, for it has implications not only for the engine design, but system performance, the Guidance, Navigation, and Control subsystem design, and mission analysis. Another example is electrical versus structural component failures. These are major design guidelines which need to be specified at the very beginning of the design process so the subsystem designers have sufficient guidance to perform their designs. Ultimately, the system designers must take a practical look at the system to determine what faults should be considered at this phase. The above ideas are just that, ideas which can be used as a springboard for further consideration for the system in question.

Some definitions for the "Root Causes" and "Temporal Manifestation" fault classes shown in Table 3.1-2 are given below.

Design

The process of converting a function into a physical or logical implementation introduces a built-in fault that manifests itself when the functionality is called upon.

Specification

The process of specifying a function introduces a built-in conceptual fault that manifests itself when the functionality is implemented and subsequently called upon.

Part Lifetime

The non-ideal nature of the materials that make up a hardware piece/part introduces a fault that manifests itself when the normal lifetime of the piece/part is reached.

Environmental

An influence external to the system (including external to the ground or commanding portion of the system) induces a fault within the system. These faults are usually caused when the system is exposed to an external factor which is outside of its designed capability.

Permanent

At some point in time, a fault occurs in a hardware element and its functionality is permanently lost. This is usually associated with, but not limited to, the random piece-part failure.

Transient

A fault which manifests itself temporarily. A transient fault is a one-time event.

Latent

A fault whose symptoms have not yet manifested themselves.

Intermittent

The same as the transient class, except that the fault is identified to come and go any number of times, and at any frequency.

Table 3.1-3 shows the basic form and options for a fault tolerance specification at the system level. Remember that the fault classes shown in **Table 3.1-2** are not a comprehensive list, but provide a starting point from which the systems requirements can be begun for SHM.

The xxx shall be able to operationally tolerate mmm faults of classes yyy.		
or		
The xxx shall be able to safely tolerate nnn faults of class yyy.		
xxx = system	nnn = number of	yyy = root cause
= function	faults which the	(design, spec,...)
= subsystem	item can tolerate	= temporal manif.
	and without loss	(perm., transient,...)
mmm = number of	of vehicle or life	= function
faults which the	(fail-safe)	(Propulsion, G&C, ...)
item can tolerate		= implementation
and still perform		(electronic, SW, ...)
its nominal		
functions		
(fail-operate)		

Table 3.1-3. Fault Tolerance Requirement

The next three sections discuss these fault classes and requirements within the context of the implementation: hardware, software, and operations, as shown in **Table 3.1-4**. The intent at this phase of the design is to specify the fault classes and types to be tolerated by the system to sufficient detail for the subsystem designer to define the subsystem concept, with a clear notion of what faults the subsystem should tolerate.

<ul style="list-style-type: none"> • Hardware <ul style="list-style-type: none"> - Design - Specification - Part Lifetime - Physical Performance - Environmental • Software <ul style="list-style-type: none"> - Design - Specification 	<ul style="list-style-type: none"> • Operations <ul style="list-style-type: none"> - Design - Specification - Physical performance - Mental performance - Malicious Intent - Environmental
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Table 3.1-4. Fault Cause Classes within Implementation

3.1.3.3.4. Hardware Requirements

As was mentioned in Section 1.2.2, system designers and specifiers usually assume that random hardware failures (those in the “permanent” and “part lifetime” fault classifications) constitute the primary cause of system failure. Again, experience demonstrates that this is rarely the case, and hardware (as well as software and operational) HM requirements should reflect this fact. In practice, faults in the hardware are usually caused by human performance faults in the manufacturing process, environmental faults, or design faults. If the system specifiers and designers get requirements such as “the system shall operationally tolerate all single-point hardware faults in the specification class” into the initial requirements, a number of things will happen. First, the subsystem designers will wail and gnash their teeth later on in the design process as they have to somehow demonstrate compliance with this requirement. This is not necessarily bad, because second, they may come up with provisions to handle the entire fault class as opposed to individually proving each fault in the specification-class list. This is good in that the final system design will potentially handle unanticipated faults within the specified classes. Third, this will cause the hardware faults identified at the increasingly more detailed levels in the design to be categorized into fault classes so that the requirement can be met. Fourth, the FMEAs generated will be more complete as they will consider other fault classes besides permanent. (The FMEAs will, by definition, contain faults from all classes explicitly referenced by the requirements.)

The list of faults classes in Table 3.1-2 represent a starting point for fault classes for specific systems. The primary issue in SHM hardware requirements is to expand the view of hardware faults from permanent part-life faults to include transients, latents, intermittents, and the faults caused in the hardware by manufacturing process errors. Including these fault classes affects the design insofar as specific built in test or other features are necessary for latent failures, software rollback or reinstatement to handle hardware transients, and possibly additional levels of fault tolerance to deal with the higher fault rates associated with these classes. The reliability analyses which occur will also be affected substantially, since the failure rates for the system will be increased.

It must also be noted that fault avoidance is a typical hardware fault strategy, with increased structural and mechanical margins levied as requirements. Those faults which are to be "avoided" must also be clearly specified, so the designer understands the failure modes under consideration for the added design margins.

3.1.3.3.5. Software Requirements

Only the fault classes of "design" and "specification" directly affect the HM requirements levied on the software aspects of system design. System designers and specifiers usually assume that software design and specification errors will be tested out of allocated functionality. Seldom do they write bold requirements like, "the system shall safely tolerate a fault in the software specification class." However, explicit requirements like this form the basis of the SHM design methodology and should make their way into the requirements at this level. (Requirements like these also create primitive, keep-alive or safing functions that keep the system going in spite of the least expected design and/or specification related failures.)

While only the fault classes of design and specification directly affect the HM requirements on the software aspects of system design, the software clearly has to deal with the errors that propagate as a result of faults within the hardware (sensors, effectors, communication devices, etc.) that the software interacts with. In the ideal world, the system specifiers and designers would construct error and fault containment regions around these hardware elements, but cost and other constraints usually prohibit this software-wise utopian situation. Thus, requirements at this and lower levels need to address hardware/software interactions that are known or anticipated.

Additionally, case after case of HM V & V points to the fact that system designers and specifiers spend too little time and effort developing the requirements for HM functionality at this level. Specifically, requirements as to what software-related functionality will be required of one subsystem as a result of an HM related request or activity by another seem the most neglected.

3.1.3.3.6. Operational Requirements

Operational requirements have been under emphasized in the development of HM systems. Human errors/human performance inadequacies can be a major influence on the dependability of a system. For example, if human performance is ignored in reliability estimates, it is assumed that operator performance is invariably perfect, which it clearly isn't. Therefore, reliability is overestimated.

Operational requirements must be developed with a clear understanding of the plausible human roles in the system. Table 3.1-5 shows what those role(s) contributes to/detracts from in the health management system.

Operator	Monitor
Maintainer	Contingency Planner
Programmer	Anomaly Resolver
Decision Maker	Commander

Table 3.1-5. Some System Human Roles

Operational requirements are often divided into tasks and personnel requirements, operator/maintenance/safety/biomedical support requirements, equipment/facility requests and human-equipment interface requirements. This breakdown stresses dealing with human fault avoidance issues and fall under the expertise of human factors engineering, maintainability engineering and safety engineering. Requirements associated with this breakdown, but dealing with human fault tolerance, has been largely ignored in the past.

For example, if a human is responsible for commanding, what are the consequences of a bad command? an inappropriate command? a miscommand?, etc. Answers to these questions may result in a requirement such as, "the system must be single fault tolerant to a command error."

3.1.3.3.7. System Reliability and Maintainability Apportionment

As shown in Table 3.1-6, a major part of Systems Engineering is allocating the overall Launch System reliability and Maintainability goals to subsystems. Apportionment includes both vehicle and ground resident subsystems. Mission reliability = Statistical Design Reliability X Process/Operations Reliability.

<u>Subsystem</u>	Mission Reliability Goal	Statistical Design Reliability Goal	Process / Operations Reliability Goal	Mean Time To Replace Module
Core Ordnance	0.9999	0.99995	0.99995	15 Min.
Core Electrical	0.9981	0.9992	0.9989	20 Min.
Avionic Checkout System	0.98	0.9899	0.99	5 Min.

Table 3.1-6. Reliability and Maintainability Apportionment

SHM personnel must actively participate in the allocation of reliability and maintainability to various subsystems. These "classical" reliability and maintainability activities need to include all fault classes indicated in the requirements, as noted in the previous sections. Thus part or component count is only one of many sources of data to consider as the apportionments are made. It should be noted that since in general data is lacking or unreliable (no pun intended), these allocations should at this phase of the design process be considered goals, not hard requirements. During conceptual design, allocations are made to the subsystem level. During preliminary and detail design, allocations of reliability and maintainability are made down to the module, box, and parts levels. At this time, allocations are considered requirements, although they are usually "provable" only by analysis, not test. Where applicable, the subelements of reliability and maintainability are allocated such as fault tolerance and fault diagnosability. In the later design phases, false alarm and false diagnosis "not to exceed" limits are also allocated.

3.1.4. SHM Initial Design Requirements Phase Summary

Summarized in **Figure 3.1-13** are the key activities of the initial requirements phase for SHM development.

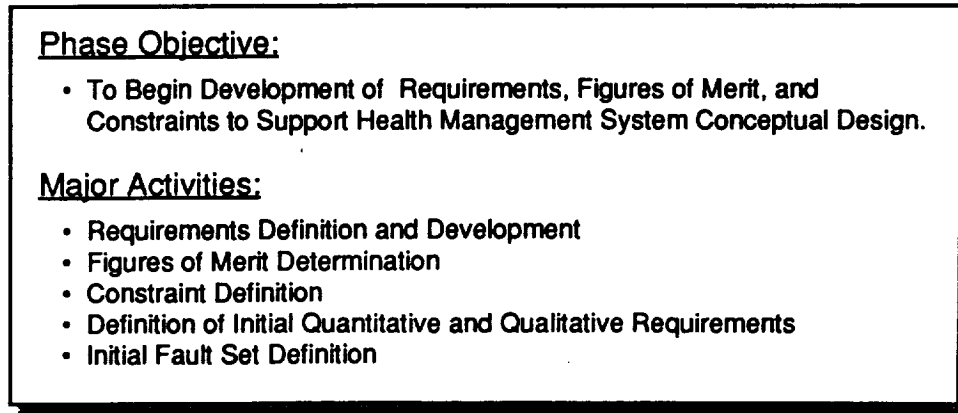


Figure 3.1-13. Requirements Phase Goals & Activities

During the system requirements phase, when the top level customer demands are determined and documented, a team approach involving the customers, users, and manufacturers is necessary to create the correct set of requirements. The QFD method is one method which has been used to perform this function. Once this has been completed, the initial system concept or concepts are generated. Generating the system concept involves various trade studies, which require figures of merit to judge the results. The QFD was used as an example, showing how the “A-3” matrix, the “house of quality”, can be used to help determine trade studies. In addition, the “A-1” matrix creates various figures of merit which can be used. The last portion of this phase is the generation of SHM requirements. These requirements include quantitative allocations of reliability and qualitative fault tolerance requirements against a specified fault set. There are tools available to aid the requirements development process.

3.2. SHM Conceptual Design Phase

This section explains the approach for development of the HMS conceptual design. The first two subsections (System Level Design Inputs to HMS design and Conceptual Design Requirements) discuss what information is utilized and expanded from the initial requirements design phase and the overall system engineering design process. The Analyses and Design subsection discusses the major analyses and design approach from the designer's perspective. The last subsection contains a discussion of the preliminary design phase requirements.

What Constitutes a Health Management System Design?

Shown in **Figure 3.2-1** is a typical systems decomposition of a launch system. The segment level decomposition is that utilized by the National Launch System project in early 1992. The health management system implementation must cover the vehicle, the ground system, and the information system segments.

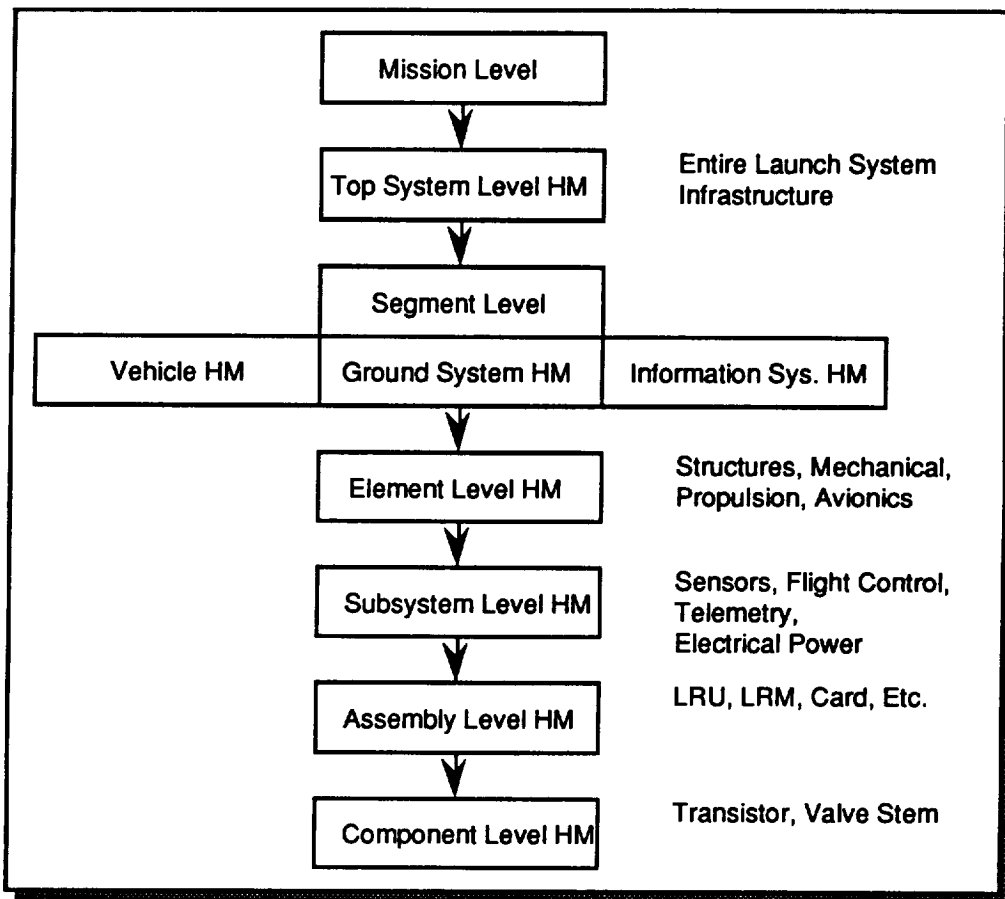


Figure 3.2-1. System Level Decomposition

At the end of conceptual design, a health management system concept should extend to the subsystem level. At the end of conceptual design, there can be a mix of singular and plural design concept(s) for SHM at the different levels. For example, in the vehicle segment, there could be an option of dedicated health management processor sets in both the core and boosters or a dedicated health management processor set just in the core of the vehicle. Another possibility is a total absence of a dedicated health management processor set at the vehicle level, with the health management totally integrated into other processor sets such as the G&C (guidance and control), engine controllers, etc.

Viewed from the very top level of launch system design, the elements described in this system health management methodology are an integral part of an overall launch system Integrated Product Development (IPD) approach. The health management system (HMS) must be developed concurrently with and integrated with the overall launch system design. This means that there are numerous HMS design tasks that overlap with traditional launch system design tasks such as avionics design, information network design, etc. The focus of this document is placed on HMS specific activities. Design to prevent and mitigate faults must be a part of every designer's philosophy. It is important to indoctrinate conceptual designers on the concepts of SHM so that provisions for health management are built in at project inception. Although SHM add-on is possible, it is more expensive and difficult.

3.2.1. Conceptual Design Phase Objective

The objective of the conceptual design phase is to generate SHM designs which take failures into account and develop requirements and design tools for the preliminary design phase. The process to define the HMS concept involves a team approach because it is not possible to cleanly separate HMS design features, hardware, and software from the overall launch system. SHM is superimposed across the entire systems design. Some HMS design features are embedded and fully integrated into the subsystems, while in other subsystems, there may be largely stand alone distributed SHM function unique hardware and software. Some subsystems will have *both* embedded and stand alone design features to implement SHM. As an example, in today's aircraft engine health monitoring systems, major control parameters often see multiple use as trended parameters for component life usage calculation and/or on-condition maintenance flags. If sufficient data and detail is available to conduct trade studies with, a single health management system approach may emerge, or possibly a favored baseline concept with options. The conceptual design for the system, and the requirements generated from the trades and analyses during this phase, must be to a level deep enough to begin the preliminary design in the next phase.

Phase Objective:

Generate Conceptual SHM System
Designs Which Take Failures Into Account
and Develop Requirements and Design
Tools for Next Design Phase

Table 3.2-1. Conceptual Design Objective

3.2.2. Conceptual Design Phase Major Activities

Shown in Table 3.2-2 are some of the products and activities which formulate a HMS conceptual design. The table illustrates that the term **HMS conceptual design or HMS architecture is more than just a parameter list**. Many design features such as containment regions, redundancy levels, design margin or special materials selection for passive fault tolerance, fault response logic, inspections, control loops, etc., and other design features contribute to the HMS architecture definition.

- **Refine Fault Set Definition for Application At Subsystem Level**
- **Generate Fault Accommodation List (Within Subsystems At Best Fidelity Known) To:**
 - Design Out (Identification of Basic Approach How)
 - Inspect Out
 - Tolerate (Identify Conceptual Active or Passive FT Approach)
 - Monitor/Report Only
- **Develop Functional Fault Matrix Identifying Top Level Fault Detection, Isolation, Response, Verification Plan**
- **Define Parameters To Monitor:**
 - Desired Parameters for Predictive Trending
 - For Active Fault Tolerance
 - Parameters for Life Usage Calculations
 - Ground Checkout Parameters
- **Identify Major Timing Constraint/Requirements (Time to Criticality)**
- **Establish Error/Fault Containment Region (ECR/FCR) Partitioning Between Subsystems and At Major Levels Within Subsystems**
- **Develop Health Management Data Handling Plan**
- **Formulate Top-Level SHM Verification and Validation Plan.**
- **Decide Upon Degree of System Autonomy**
 - Ground Based Vs. Flight Based Decision Partitioning, Degree of Human Role In SHM Functions
- **Input SHM Conceptual Design Software Requirements to Overall Vehicle Software Plan/Requirements**
- **Identify Passive Fault Tolerant Concepts and Make Appropriate Up Front Design Provisions**
 - Margins, Special Materials for Delayed Failure, Fall On Selected Paths, Physical Containment and Separation
- **Develop Requirements for Simulation/Testbed Design**
- **Refine Analysis Tools for Design Support**
- **Develop SHM Requirements for Preliminary Design Phase**

Table 3.2-2. Conceptual Design Activities and Products

3.2.2.1 Phase Activities Flow

Figure 3.2-2 is a flowchart showing the overall SHM conceptual design approach from a designers perspective. This flowchart describes a systematic approach to assure that the entire integrated system and the individual subsystems are designed to accommodate fault and errors that arise from many fault domains. It is especially important to consider system tolerance to faults/errors very early in the design process to avoid a “locked in design”, with inability to make design provisions which can only be effectively inserted early in the design process. Since many faults and errors must be handled at the system’s level, an overview of the entire system from a fault/error perspective in conceptual design may be the most important step of this methodology.

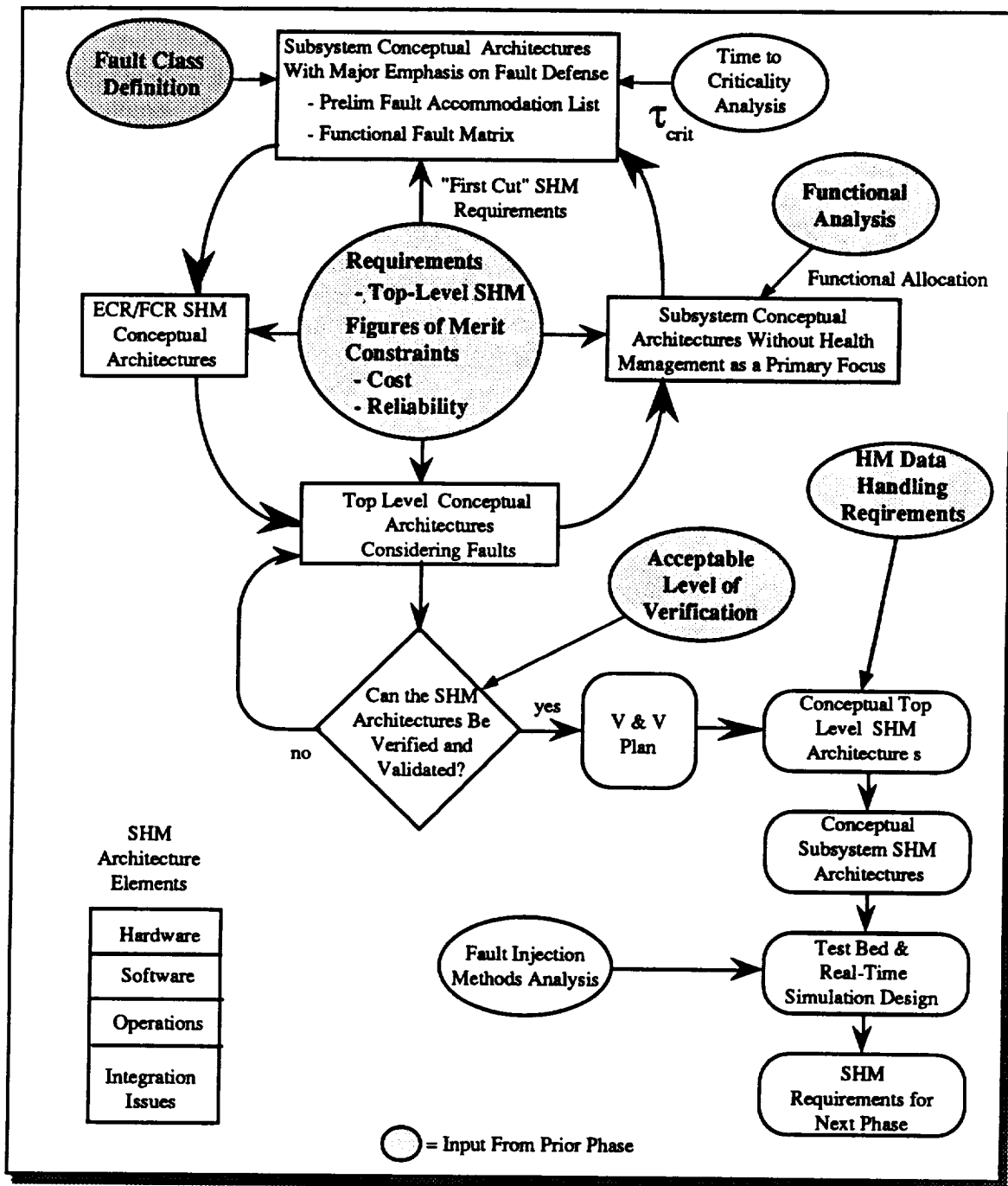


Figure 3.2-2. SHM Conceptual Design Flow

Requirements, figures of merit, and constraints are shown as the “hub” of the design approach wheel of the Figure 3.2-2, for these elements interplay with the entire design process. Major inputs from the initial requirements phase (prior phase) are shaded. The design approach wheel consists of five parts: subsystem conceptual architectures without health management as a primary focus, subsystem conceptual architectures with major emphasis on fault defense, ECR/FCR conceptual architectures, Top-level conceptual architectures considering faults, and V&V. The products from this design phase are the top-level V&V plan, conceptual top level architectures, conceptual subsystem SHM architectures, testbed & real-time simulation design and the SHM requirements for the next phase.

The goal of the SHM design process is to inject design provisions into the launch system conceptual design to achieve high dependability in a cost effective, systematic manner. The major elements of the SHM conceptual design flowchart are detailed in Section 3.2.3.

3.2.2.2. Major Analyses

A short description of the major analyses of SHM conceptual design are as follows:

3.2.2.2.1. Time to Criticality Analysis

Time to criticality analyses occur at several levels within this methodology. In conceptual design, the general functions during the major vehicle operational phases are examined for timing limits for execution of compensatory action in failure scenarios (see Section 3.2.3.2.1).

3.2.2.2.2. Fault Set Refinement Analysis

The faults of the major subsystems are identified relative to the fault classifications established. An emphasis is placed on identification of subsystem interaction, human error, Byzantine, and many other types of faults neglected in most FMEAs (see Sections 3.2.3.2.2 and 3.2.3.2.3).

3.2.2.2.3. Fault Injection Methods Analysis

This methodology places a major emphasis upon verifying the ability of the system to tolerate faults, hence, major efforts are expended early in design to demonstrate system effectiveness for fault tolerance (see Sections 3.2.3.2.7 and 3.2.3.3.1).

3.2.3. Conceptual Design Development Approach

This section details the tasks performed for SHM during the conceptual design phase.

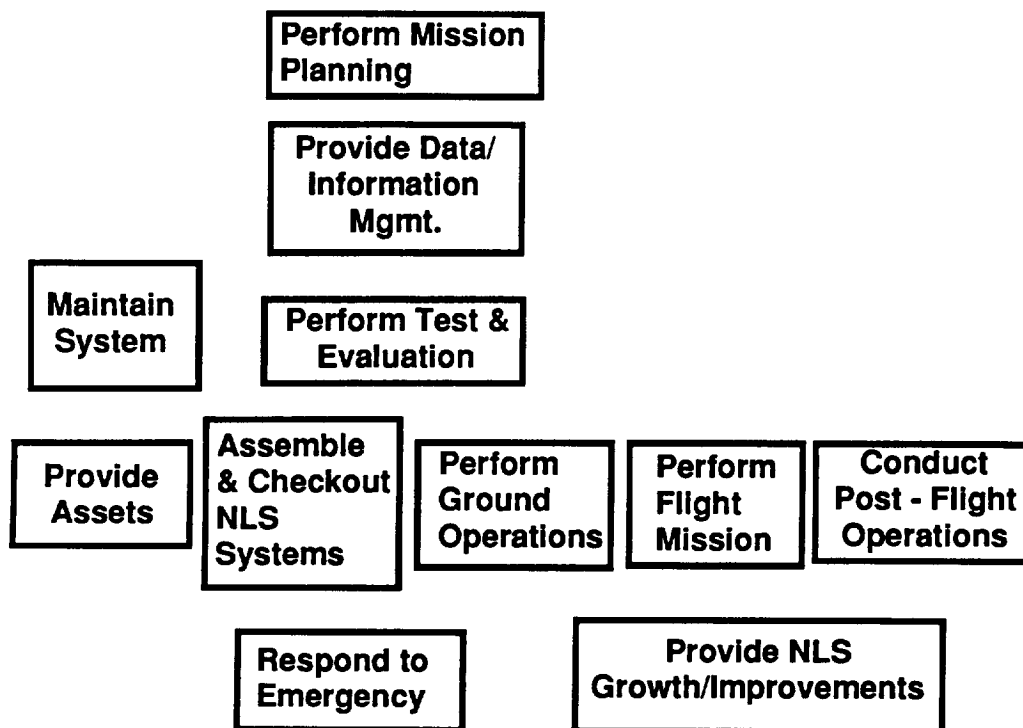
3.2.3.1. System Level Design Inputs to HMS Design

3.2.3.1.1. Subsystem Concepts Without Health Management As A Primary Focus

Figure 3.1-2 showed that top level functional analysis and system concept development were considered as a part of the initial requirements phase. In conceptual design, the functional analysis must be driven down one or more tiers to support design to the subsystem level. At this point, the focus is still primarily on subsystem synthesis with respect to the primary subsystem functional requirements. The design team is focused on the total system objectives, and cognizant but not sharply focused towards the fault/error perspective of design.

An example of the elements of very top level functional chart for the National Launch System is shown in Figure 3.2.-3. For simplicity, the interconnecting gates and arrows between the elements are not shown. From this chart, functional blocks are decomposed into subfunctions. Typically, several tiers of functional decomposition are required to move to a level of functional definition sufficient to support subsystem design. From the viewpoint of SHM, there is nothing particularly specific to SHM which must be considered, other than simply being part of the standard process. The one unique item during this phase is discussed in the next section.

• **Health Management Contributes To Almost All of the NLS Top Level Functions**



FUNCTION 0.0 PERFORM NLS FUNCTIONS

Figure 3.2-3. Main Elements - NLS Top Level Functional Flow

3.2.3.1.2. Subsystem Design and Figures of Merit

To support subsystem conceptual design, the top level figures of merit (such as reliability and life cycle cost) shown in the center of the conceptual design wheel of Figure 3.2-2 remain the same. However, the figures of merit are better decomposed into their constituent elements so that design concepts can be quantified and weighed against one another. As an example, the attribute of maintainability might be decomposed into fault diagnosability, accessibility, repairability, and supportability. These attributes can in turn be described by groups of quantifiable parameters, or lower order "figures of merit". The top level figure of merit has now been sufficiently decomposed in constituent elements which are measurable (for more see Section 3.2.3.2.3).

The subsystem design concepts created from the lower tier functional breakouts rapidly expand the system definition and stir the excitement of the more specialized engineering design teams. A great many trade studies are identified at this point in design, and there is a great need to identify trades which interact, and hence require a true systems engineering perspective. There is great danger in adding together subsystem level trade studies without a top level systems perspective. Figures of merit can help balance the trade studies across the subsystems.

3.2.3.2. Conceptual Design Requirements

After substantial progress is made in launch system subsystem conceptual design, "first cut" HMS requirements are formulated which enable system and vehicle health management concept synthesis to begin. This concept is illustrated in Figure 3.2-4. Without HMS requirements to an appropriate level of fidelity, the designer cannot progress well in the formulation of the conceptual health management design. From the perspective of the conceptual design flowchart of Figure 3.2-2, the first cut HMS requirements enable the block titled "subsystem conceptual architectures with major emphasis on fault defense" to be implemented.

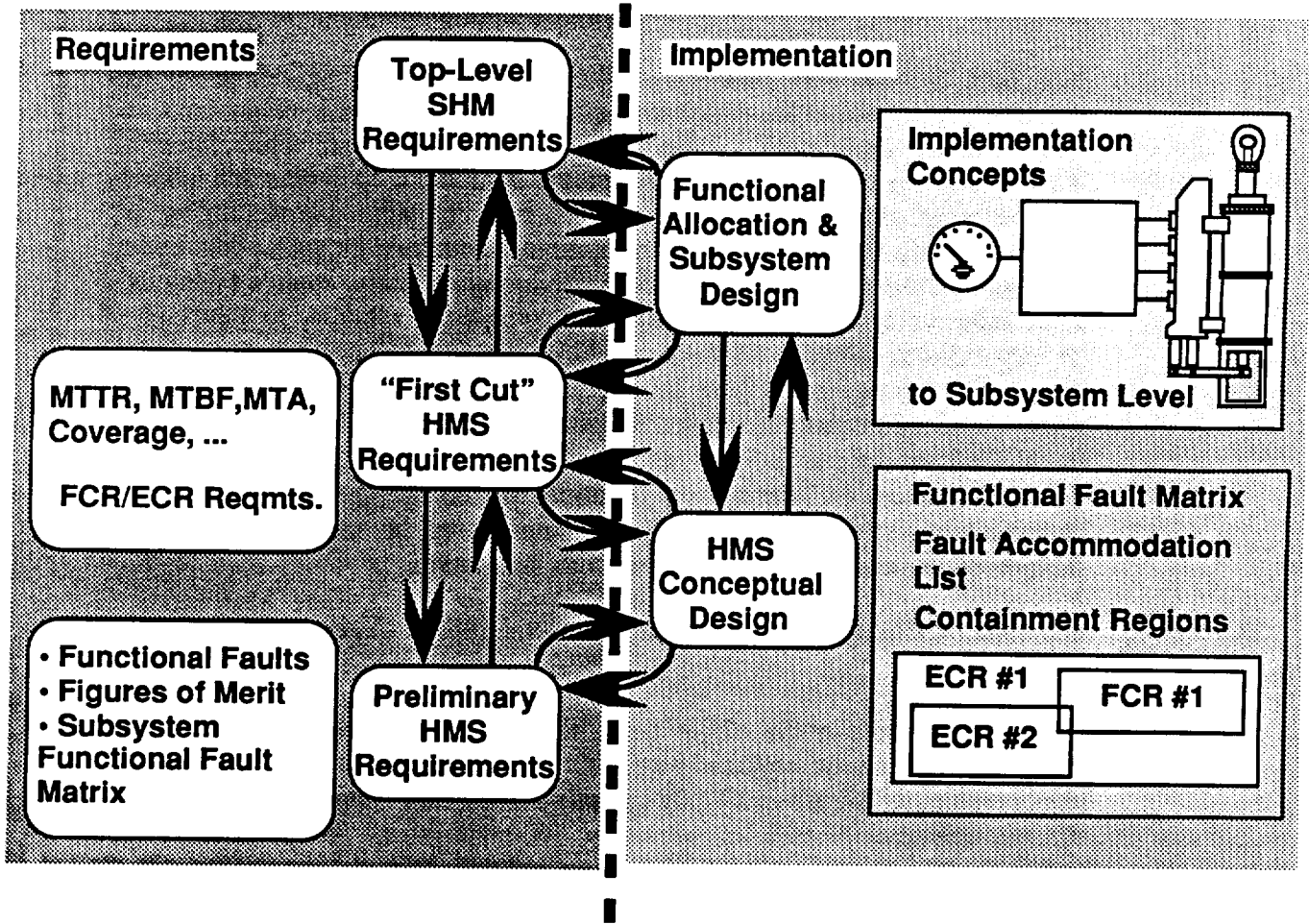


Figure 3.2-4. Requirements Relationship to Conceptual Design Phase Flow

3.2.3.2.1. Subsystem Time to Criticality Requirements

The time to criticality analysis of the previous section defines a set of criticality levels and times which are associated with particular subsystems in each phase of the mission. These times represent the time in which failures within that subsystem must be detected and responded to before the failure compromises or destroys the mission or vehicle (when the criticality is sufficiently high). The point of this analysis is to levy requirements upon particular subsystems based upon the analysis. Specifically, when a failure of a particular subsystem causes the loss of the vehicle or mission, the fault must be responded to before the time to criticality associated with the fault expires. This is the timing requirement to be levied upon the subsystem: The xxx subsystem shall respond to internal faults of classes a, b, and c (as determined by the fault set defined earlier) within time t. These requirements form one of the basic pieces of information required by the subsystem designer to determine an appropriate HMS design, for it states the basic time relating failures of the subsystem to effects on other subsystems.

3.2.3.2.2. GIMADS Program Contribution To SHM Conceptual Design Requirements

One of the goals of the SHM methodology is to uncover faults in the inspection and checkout phase before the flight. The dependability attribute tree presented as Figure 3.1-11 has a branch titled "fault diagnosability" under the maintenance branch of the tree. Testability/fault diagnosability is extremely important to uncover faults in a timely fashion before reaching the launch pad, where maintenance costs rise dramatically. Those responsible for SHM must work closely with the assembly and operations teams to make sure the assembly and checkout procedure is understood and appropriate testability requirements levied upon vehicle and ground systems. As suggested by the dependability attribute tree, accessibility, repairability, supportability, and many complex attributes interact to impact operability and throughput of the launch complex.

The Air Force Generic Integrated Maintenance and Diagnostics (GIMADS) program can provide useful guidance to the health management system design process in the area of testability and integrated diagnostics. The Aeronautical Systems Division of the United States Air Force at WPAFB awarded the first contract of the GIMADS program in February 1987. GIMADS will finish in 1992. GIMADS has focused on improving the process for getting integrated diagnostics into future Air Force Weapons Systems.

The new GIMADS generated MIL-STD-1814 has focused on how to get integrated diagnostics encapsulated within the requirements for a system. The companion GIMADS document is Air Force Guide Specification AFGS-87256. The breadth of MIL-STD-1814 is such that ideas and recommendations within it should be reviewed for incorporation within both SHM requirements development, and on a broader scale, the vehicle Systems Engineering Management Plan. Some specific suggested GIMADS contributions to the "first cut" SHM requirements are shown in Figure 3.2-5. GIMADS recommends that the maintenance and operation plans for vehicles be analyzed for decisions and events that require diagnostic information. Diagnostic requirements are then generated from the maintenance and operations plans. The relative immaturity of the maintenance and operations plans during the SHM initial requirements phase will provide information for only top level diagnostics requirements during this phase of SHM development. As the operations and maintenance plans evolve, more detailed SHM requirements will develop.

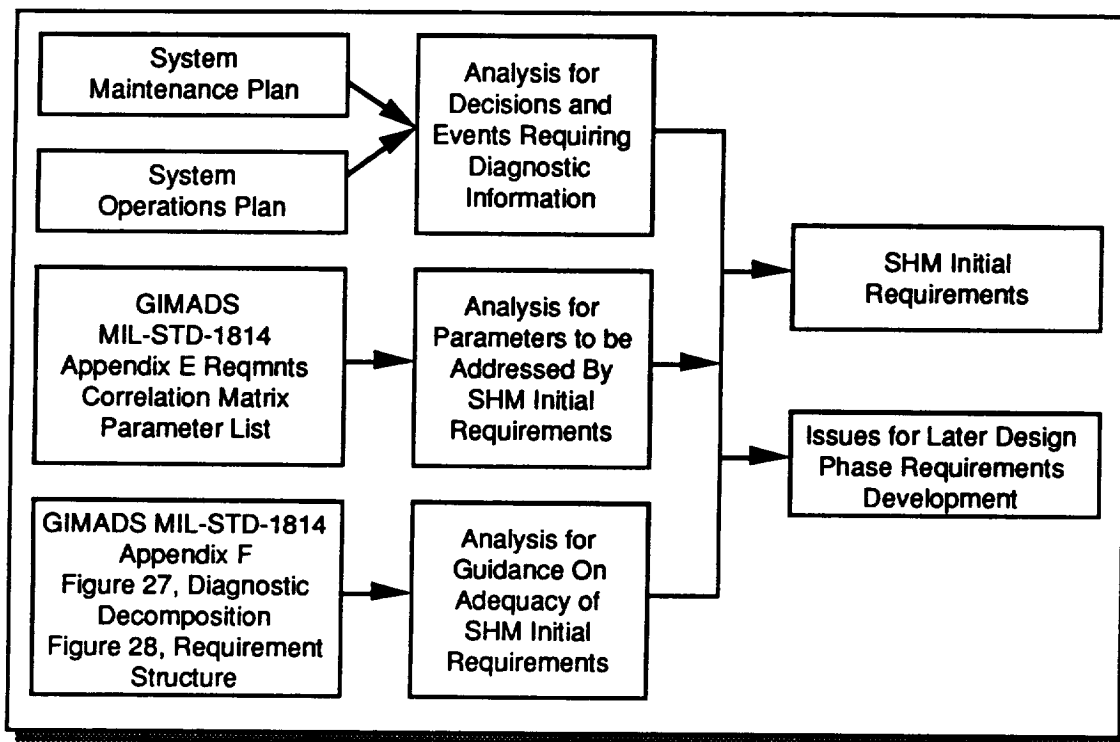


Figure 3.2-5. GIMADS Contributions to SHM Requirements Development

Appendix E of GIMADS developed MIL-STD-1814 lists 27 requirements correlation matrix parameters which drive system level diagnostics requirements. Although the list was developed for reusable aircraft, many of the diagnostic related parameters of the Appendix are launch vehicle applicable. It is recommended that the system level requirements list of GIMADS MIL-STD-1814 Appendix E be incorporated in the SHM initial requirements development process. Some of the more important diagnostic related requirements noted by MIL-STD-1814 are acceptable levels of false detections and isolations, diagnostic human factors design criteria, mean time to diagnose, diagnostic manpower, fault reporting latency, and others.

Figure 27 of Appendix F of MIL-STD-1814 provides valuable guidance for generation of requirements that influence the operations plan, the maintenance plan, and the health management concept. This figure and accompanying GIMADS text also provides guidance in development and identification of constraints impacting the SHM design. Figure 28 of MIL-STD-1814 Appendix F is also useful for developing a diagnostic requirements structure.

There are other useful elements of GIMADS valuable to the development of launch vehicle diagnostic requirements. MIL-STD-1814 should be used as a valuable technical source for requirements development.

3.3.3.2.3. Dependable System Attribute Parameter Formulation and Allocation

As the subsystem implementations solidify, the health management requirements gain more depth (recall the requirements-implementation iteration figures previously introduced such as Figure 3.2-4). The dependable system attribute tree introduced in the initial requirements section (Figure 3.1-11) is applied to specific subsystem concepts. The dependable system attributes are now expressed as quantified parameters and levied/allocated as requirements upon the subsystems. Some examples of parameters associated with dependable system attributes are shown in Figure 3.2-6.

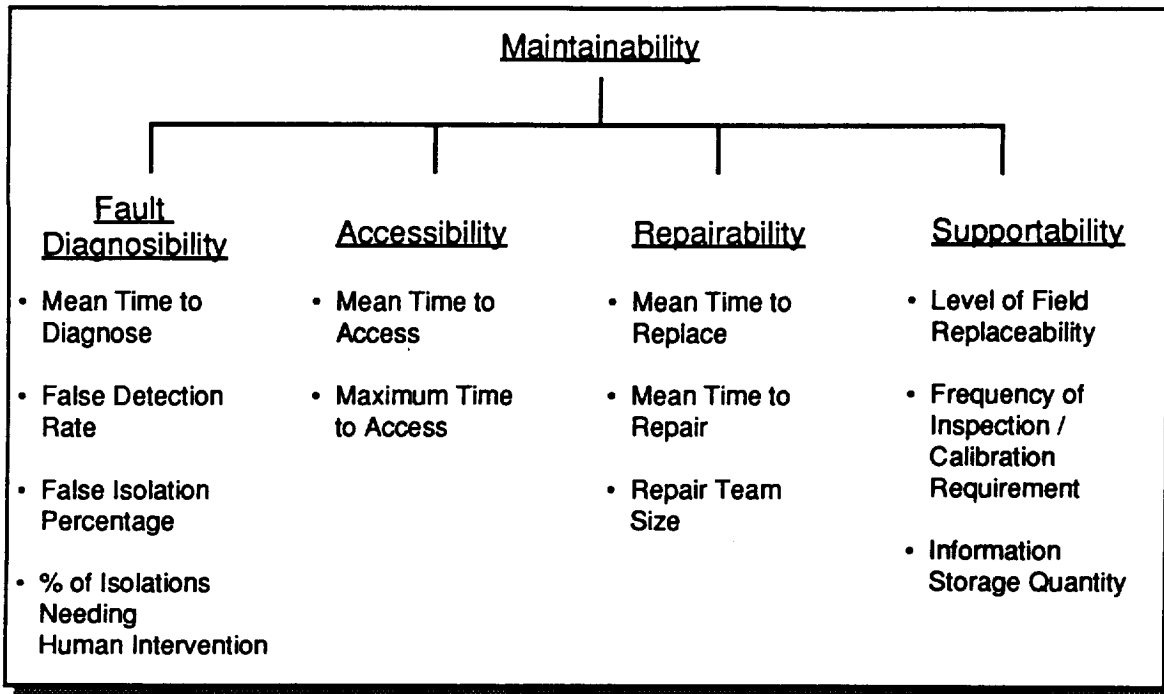


Figure 3.2-6. Maintainability Branch of Dependable System Attribute Tree

3.3.3.2.4. Performance, Safety, and Other Requirements for SHM Conceptual Design

In addition to requirements related to reliability and maintainability, requirements from performance, safety, and other considerations impact the SHM design. **Table 3.2-3** lists typical justifications for systems health management. More detailed requirements are likely to be derived from each of these initial requirements categories as the SHM design proceeds.

Preflight

- Supports Faster and More Comprehensive Ground Checkout
- Improved Ascent Reliability Via More Comprehensive Holdown Firing Check

Flight

- Improved Flight Mode Failure Prevention Via Fault Tolerance Including Reconfigurability

Post-Flight

- Reduced Downtime After Failures/Incidents Through Expedited Availability of More Comprehensive Information
- Improved Safety and Reliability Through Better Post-Flight Data Analysis and Environment Confirmation

Table 3.2-3. HM Benefits for Launch Vehicles

3.2.3.2.5. SHM Software Requirements Development

Software has been a neglected aspect of SHM design. Experience has indicated that up to 75% of health monitoring system cost could be software. Hence, the software part of SHM must be given at least equal status with hardware in the requirements development and the other phases of SHM design.

Software should buy its way into a SHM design just as hardware on a cost effectiveness basis. In general, software is more difficult to analyze for cost effectiveness because of its less tangible nature, a lack of well recognized and accepted tools to estimate software cost and complexity, and a tendency for software to stay in a state of uncertainty and flux until well into the detail design phase. SHM software is a very immature technology area, at least from the perspectives of cost effectiveness modeling, automated code generation, and well understood and accepted application methodologies.

Shown in Table 3.2-4 are typical categorizations of software requirements. During the functional flow diagram construction and subsequent requirements allocation, there will often be options on the degree of hardware, software, and human elements utilized for function accomplishment. Experience has indicated that software solutions provide more flexibility for future modification. Since a significant percentage of launch vehicle failures are attributable to human error, coded techniques which eliminate possibility for human error are attractive if verification techniques for the software can assure latent and more visible software errors are extremely rare.

Functional Requirement

A Requirement That Specifies Functions That a System Or System Components Must Be Able of Performing.

Performance Requirement

A Requirement that Specifies Speed, Accuracy, Frequency , and Other Performance Characteristics That a System or System Component Must Possess.

External Interface Requirement

Specification of Hardware, Software, or Data Base Elements With Which A System or System Elements Must Interface, or Sets Forth Constraints on Formats, Timing, or Other Factors Caused By Such An Interface.

Design Constraints

A Requirement That Affects or Constrains the Design of the Software System or Software System Component. Examples: Language Requirements, Physical Hardware Requirements, Software Development Standards, Software QA Standards.

Quality Attributes

A Requirement That Specifies the Degree to Which Software Possesses Attributes That Affect Quality (Portability, Correctness, Reliability, Etc.).

Table 3.2-4. Software Requirements Categories

There is no unique or singularly accepted approach for a software requirements methodology. Some of the common software requirements methodology classifications are shown in Figure 3.2-7. Perhaps the most important issue is to select a software requirements methodology consistent with the organizational structure, personnel, and SHM project team. There are two basic categorizations of software requirements methodologies, the document driven and the process driven approach. The IEEE Guide to Software Requirements Specification is an example of the document driven approach, while DOD-STD-2167A is an example of the process driven approach. The oldest approaches are the structured methodologies, with the object oriented methodologies perhaps the most rapidly evolving.

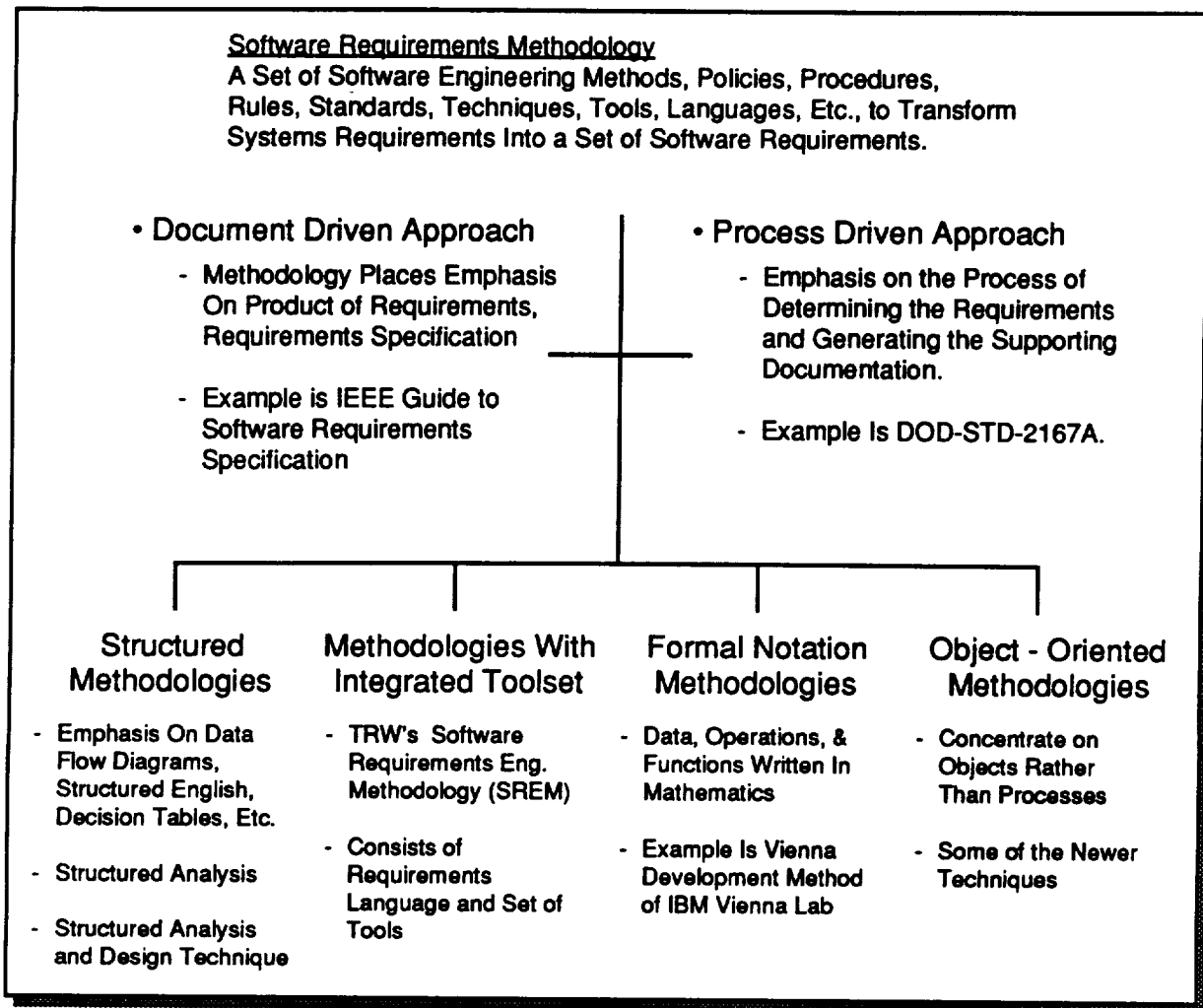


Figure 3.2-7. Software Requirements Methodology Classifications

Formal notation methodologies are a relatively new approach to design and verification of complex systems. The Airbus aircraft flight control systems have extensively utilized formal notation methodologies to gain confidence in system fault coverage. Formal methods development as applicable to SHM software is an area ripe for further research, with a large payoff in the overall dependability of systems.

Our recommendation for software requirements for SHM is to survey the spectrum of software requirements methodologies available, and utilize an approach best suited to the experience base and the team composition selected for the SHM software development. Software should be evaluated for cost effectiveness just as SHM hardware is. Software must likewise be evaluated for quality as well as hardware. At the present time, the best means to assure software requirements help achieve software quality and cost effectiveness goals is still an active area of research.

3.2.3.2.6. SHM Operational Requirements

As mentioned in Section 3.1.3.3.6, human/system interaction issues have mostly dealt with human error avoidance through appropriate application of Human Factors design engineering principles. A typical HF program will address: environmental compatibility (life support and protection systems, occupational safety and health issues), input-output requirements (human abilities and limitation issues), information processing (memory, attention, workload, monitoring, control, decision making and problem solving), and authority/responsibility (human and/or machine tasks). The HF specialist will seek to reduce error to a minimum and if an error occurs, they seek to understand and to eliminate that error by modifying the human-system interaction which produced it. This is only part of the story. Dependable design requires that designers first ask how can humans introduce error into the system and can that error be mitigated or tolerated. Secondly, what are the role/roles they are going to play in the overall SHM design (fault detection, fault isolation/diagnosis, fault response/fault repair, fault recovery/safing, monitoring, test, checkout etc.).

Consideration of operational issues and human error/fault tolerance in design starts in this phase and is discussed in Section 3.2.3.2. In order to address the second question mentioned above, a SHM functional allocation must be performed. This analysis is best performed by a team of individuals consisting of a SHM specialist, one or more designers, a human factors specialist, a maintainability specialist and a safety specialist.

SHM functional allocation analysis:

- 1) Determine "first cut" SHM requirements
- 2) Determine SHM functions
- 3) List and describe all the possible ways that the SHM functions could be implemented (use system functional allocation to help decrease list). Consider that the system functions could be implemented: largely by operator personnel, human-hardware/software mix, or hardware/software primarily. Consider that however implemented will performance satisfy the system. Consider what will the system demand of the human and can the human satisfy the system demand.

- 4) Determine weight selection criteria that will be used to choose between alternatives such as cost, reliability, number of personnel, etc.
- 5) Perform a comparison between alternatives
- 6) Allocate those functions to humans and/or hardware and/or software as appropriate
- 7) Identify the man-machine interface required and operator performance criteria: frequency of required outputs, speed of required outputs, physical requirements (e.g. strength, sensory discrimination capability, decision-making capability) for implementing the function, and accuracy of required outputs.

3.2.3.2. Analyses & Design

The SHM conceptual design process follows the flowchart introduced as Figure 3.2-2 to produce the products and analyses summarized in Table 3.2-2. The flowchart is simplified, and all the interactions not shown. Many of the analyses are somewhat concurrent and interactive.

3.2.3.2.1. Time to Criticality Analysis

Definition and Purpose of Time to Criticality

Time to criticality analyses reveal the behavior of system elements as they interact with each other under fault conditions. The key features to these analyses, as indicated in the name, are the aspects of timing and criticality. Given a fault scenario, what is the effect of the fault on the rest of the system, and how long does it take for the system to get to this state if the fault is uncompensated? These analyses recur at all levels of design, starting with a top level assessment of basic system functional interactions, and continuing to a detailed component interaction level within detail design. From an SHM perspective, timing requirements and constraints which result from these analyses have a major influence on the selection of fault prediction, detection, isolation, and response stratagems.

An example of the importance of application of time to criticality analysis is reflected by a lesson learned from the Magellan spacecraft. A pictorial schematic of the Magellan attitude control system and electrical power system is shown in Figure 3.2-8. The cycle time of the Command and Data Subsystem (CDS) was 1.5 Hz, or exceeded 667 milliseconds, whereas the Attitude and Articulation Control Subsystem (AACS) cycle time was 33 milliseconds. During those portions of the Magellan mission where critical maneuvers were underway, various attitude control faults, if undetected and or uncompensated could cause loss of the mission (due to loss of control of the vehicle) within less than 100 milliseconds in the worst case, and certainly in less than 667 milliseconds in most cases. The only fix to various AACS faults was to switch components via the CDS. Since the response could not be activated and completed quickly enough to switch the faulty component out before vehicle control was lost, the system had built into its architecture an automatic single point of failure simply due to the architecture. This weakness would have been uncovered by a top level time to criticality analysis early in the design, when architectural changes were feasible. As it was, problems such as these were often found too late in the design to be fixed, simply due to cost and schedule issues. When the mission flew, the designers simply had to hope that these failures did not occur. Fortunately, these particular faults did not.

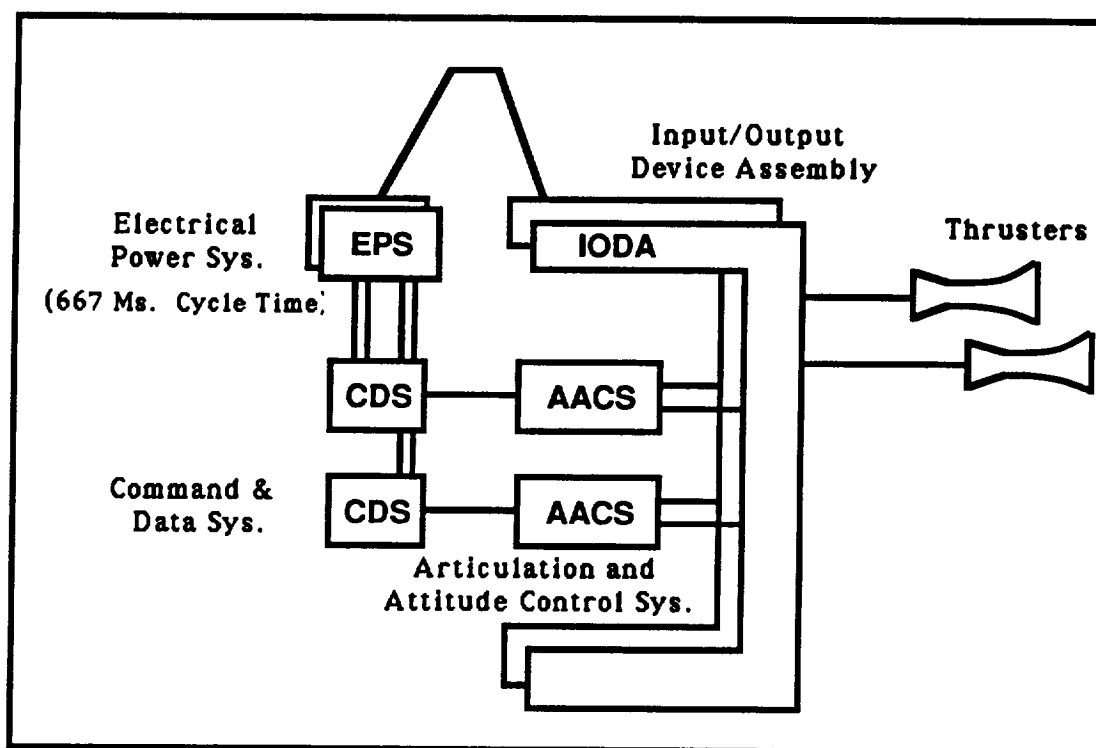


Figure 3.2-8. Magellan EPS - AACS Interaction

In a broader context, time to criticality concepts apply to almost all projects involving systems engineering and/or fault management.

Top Level Time to Criticality Analysis

The top level time to criticality analysis should begin at the system functional level. As shown in **Figure 3.2-9**, basic system functions should be identified, and mapped against mission phases. This is necessary because the fault effect and timing changes with the mission phase and function. This matrix format is sometimes described as the phase and function vector space. The phases are further decomposed into subphases or mission event intervals.

		Phase			
		Pad Operations	Holddown	1st Stage Ascent	Staging
Function	Guidance				
	Control				
	Pyrotechnics				
	Telemetry				
	Other			$\tau = 3 \text{ sec.}$ Loss of Mission	

For Each Function of Each Mission Phase:

- Time to Criticality For Each Applicable Category of Criticality
- Uncertainties and Assumptions for Worst Case Analysis of Time to Criticality
- Further Analysis Needs Identified
 - Most Probable Time to Criticality Vs. Stacked Worst Case Time to Criticality

Figure 3.2-9. A Top-Level Time to Criticality Matrix

An example of a decomposition of the liftoff through return phase to a rough subphase level is shown in **Figure 3.2-10**. This figure uses a National Launch System / Cargo Transfer Vehicle to Space Station mission event sequence as an illustration.

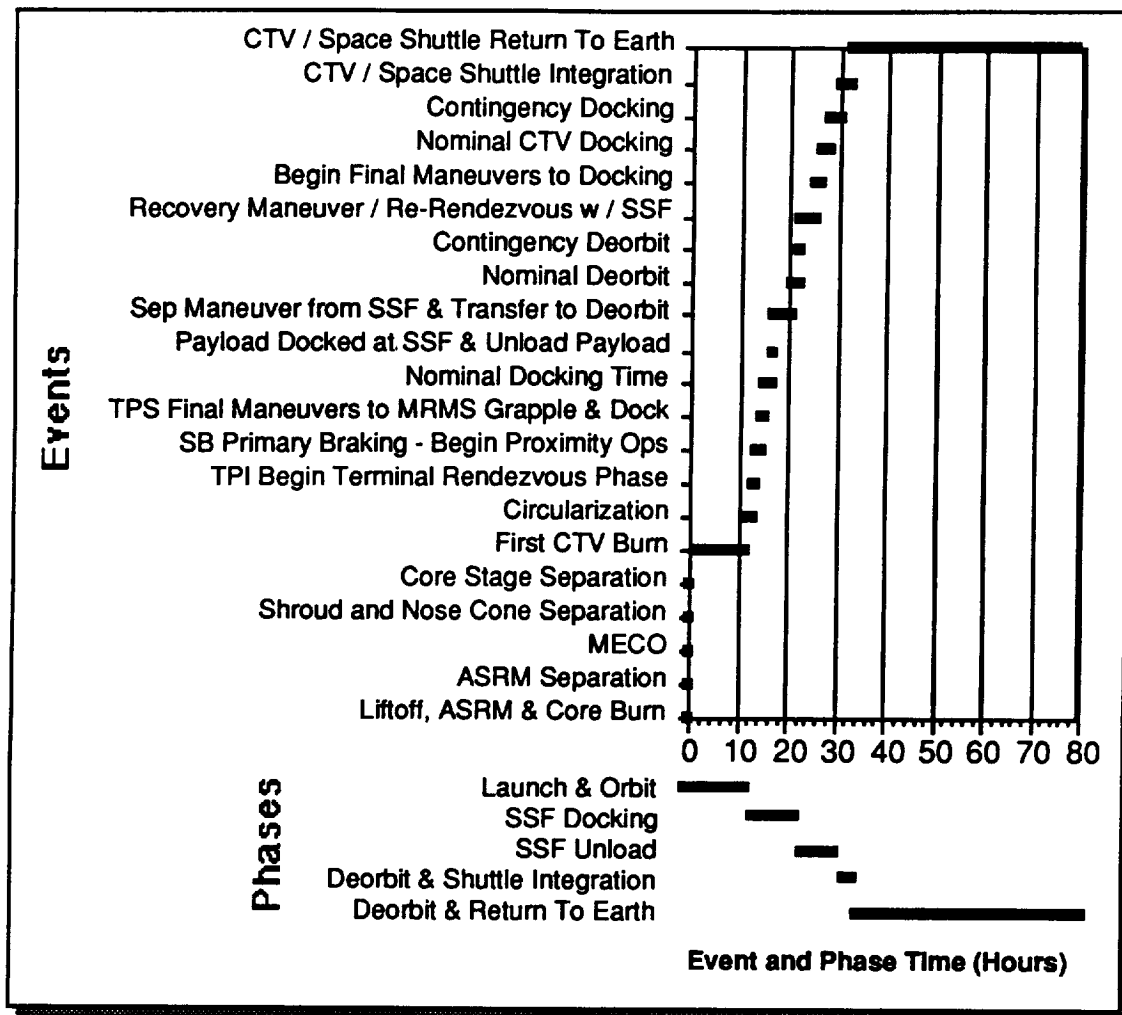


Figure 3.2-10. NLS/CTV Mission Event Sequence

The basic system functions required to execute each mission event are now identified. One at a time, each function is assumed to fail, and an estimate (to the best level of fidelity possible at this point in design) of the amount of time the system can tolerate the loss of each function before reaching different levels of criticality determined. For example, postulate the loss of vehicle power. It's effect on the structure is nil, but its effect on the control function is critical within milliseconds, or perhaps even microseconds, because the computers which process the control data will lose their memories. Thus the failure of one function is assessed versus each other function in the system for a given mission event. Two pieces of information are generated, the criticality level, and the amount of time to reach that criticality level. Top level criticality categories are:

1. Loss of life
2. Loss of mission
3. Compromised mission
4. Cost Risk
5. No Effect

These criticality categories are not the same as typically used for NASA or Air Force systems. That is intentional in this case. When this analysis was considered for the first time, and was applied to different subsystems, these particular categories were consistent with each other, and with the intent of the analysis. Other typical categorizations of criticality turned out to be not useful for this particular analysis, although they have validity for other purposes. Criticality categories may have to be customized for different types of systems.

Worst Case and Most Probable Case Time to Criticality

There are often different times required to reach different levels of criticality. A worst case analysis should be utilized for the first calculation of time to criticality, and the assumptions documented for easy review and retrieval. In cases where there are a number of “stacked” assumptions which combine in an additive fashion to generate the time to criticality, a set of worst case assumptions, a very unrealistic time-to-criticality emerges, and it may be more useful to use the “most probable” assumptions. Stacking the most probable assumptions provides the most probable time to criticality, which is a much more realistic scenario. If the worst case and most probable case time to criticality numbers significantly differ, more refined analyses are merited to assess which figures (usually some compromise) should be used.

3.2.3.2.2. Conceptual Design Preliminary Fault Accommodation List

At this point in the design process, we move around the design wheel shown in **Figure 3.2-2** to the process block titled “Subsystem Conceptual Architectures With A Major Emphasis On Fault Defense”. The layered defense approach to System Health Management was discussed in **Section 1.1** and is illustrated in **Figure 1.1-1**. As the vehicle subsystems are defined, a preliminary fault accommodation list is generated for each (see **Table 3.2-5**). This list is generated from a refinement of the fault class definitions developed in the prior design phase, preliminary FMEA information and from the best quantitative information available to predict failure rates. Where data is lacking or high levels of failure rate uncertainty exist, notes, and possibly an extra column with a mean, high, and low value of failure rate are entered. This table will be refined and expanded upon as the later design phases occur.

The level of fidelity available at conceptual design will typically vary greatly from subsystem to subsystem. It is not atypical for the propulsion system and certain avionics boxes to be well defined with detailed fault lists while other subsystem elements are only roughly conceptualized.

Subassembly or Component	Failure Modes	Detailed Breakdown of Failure Criticality, Compensability, Etc.	Pred. Rate of Occurrence
_____	_____	Critical, Non-Compensable	5 PPM
	_____	Critical, Compensable	50 PPM
	_____	Non-Critical	300 PPM

Subassembly or Component	Failure Modes	Design Out	Inspect, Check and Test Out. Also Use Prev. Maint.	Active/Passive Fault Tolerance	Detect & Record or Report Only
_____	_____	X	X		X
	_____	X	X	X	
	_____	X	X		X

- Failure Modes, Especially the Critical Ones, Are Investigated For Compensability
- Quantitative Estimates of Failure Rates Help Determine Resource Allocation and the Degree of Measures In the Design Out, Inspect and Test Out, and Fault Tolerance Categories

Table 3.2-5. Preliminary Fault Accommodation List

For each fault, mitigation techniques of (1) design out, (2) test and inspect out, (3) active and passive fault tolerance, and (4) data recording for post event data reconstruction are identified and entered in the fault accommodation list. (Recall Figure 1.1-1). The quantitative estimates of failure frequency and criticality provide an indicator as to the amount of investment merited in the four fault mitigation categories. Time to criticality analyses help define which faults are potentially compensable from a timing perspective. In the parameter selection section of preliminary design, more detail is provided on cost effectiveness assessment of fault mitigation techniques.

3.2.3.2.3. Functional Fault Matrix

The next design step is to expand the column of the **Table 3.2-5** fault accommodation list titled "active/passive fault tolerance".

Subsystem Functional and Hardware Fault Matrices

Now that there exist functional diagrams and hardware/software implementation concepts to the subsystem level, as well as preliminary failure modes and effects analyses for the various subsystems, and the subsystem-subsystem interactions at the top system level are studied. This analysis is a top down "deductive analysis" based upon assuming failures of functions within subsystems, and then investigating the result of these faults both inside and outside the subsystem. In the process of performing this deductive analysis, information regarding significant data to monitor for fault detection, proposed techniques for containing the fault, and recovering from the failure are generated, and can be captured in such a matrix. It is especially important not to overlook interaction faults at all interfaces covered by an interface control document. (More emphasis is again placed on interface/interaction faults in an expansion of the fault accommodation list in preliminary design). The top level time to criticality matrix introduced in **Figure 3.2-9** can be updated, and improved in fidelity. In turn, a lower tier subsystem functional fault matrix can be generated from the first matrix. Shown in **Figure 3.2-11** is an example of a functional fault matrix.

Func Error (Symptom)	Fault Class	Error Containment	Fault Containment	Detection	Isolation	Response	Recovery	Validation	Reporting	Variable With Mission?
Bus Voltage Outside Valid Range	Permanent	Boundary: Power Bus Mechanism: Power Bus Converter	Boundary: Power Bus Converter Output Mechanism: electrical isolation	Bus Voltage Value exceeds threshold	Bus Voltage Sensor vs Converter Output Voltage Sensor	Switch to Secondary Converter		HW injection via breadboard	Bus Voltage and Output Converter Voltage	No
Bus Voltage Outside Valid Range	Transient	Boundary: Power Bus Mechanism: Power Bus Filter	Boundary: Power Bus Filter Output Mechanism: Power Bus Filter Capacitors	Bus Voltage exceeds threshold	Bus Voltage Sensor vs Load Input Voltage Sensor	Provide Energy Storage or Absorption		HW Injection via breadboard	Bus Voltage and Load Input Voltage	No
Large IMU Error-Control	Permanent	Boundary: IMU Processor Mechanism: Reasonableness Check, Parity Residual	Boundary: Sensor / Processor I/F Mechanism: Optical Isolation	Large Parity Residual, threshold exceedance	Parity Residual & threshold exceedance for sensor	Remove sensor from list via SW		Large change to bias or scale factor	Sensor Parity Residual	Yes
Small IMU Error-Nav	Permanent	Boundary: IMU Processor Mechanism: Reasonableness Check, Parity Residual	Boundary: Sensor / Processor I/F Mechanism: Optical Isolation	Parity Residual grows over time	Parity Residual	Remove Sensor from list after period of time via SW		Small change to bias or scale factor	Sensor Parity Residual	Yes

Figure 3.2-11. Functional Fault Matrix

Use of a functional fault matrix is based upon the need for a top down analysis of failures when the low level details of the actual design are unavailable. The information generated from the functional fault analysis is then used to help select appropriate designs. For subsystems such as the engine, where the hardware implementation is generally known with great detail, it is more profitable to move directly to a hardware fault matrix based on a known design which is similar to the functional fault matrix but based on a hardware instead of a functional fault. Hardware FMEAs and hardware fault matrices are much better understood by the engine community, and represent a common language well understood and uniform within that community.

For a process designer, a computer system designer, and many avionic functions, the subsystem functional fault matrix is desired, with the preliminary FMEA following at a later stage. The advantage to thinking functionally is that a higher level perspective is invoked before locking into a specific method of design implementation. This can help the designer envision design alternatives.

Use of the Functional Fault Matrix

The functional fault matrix has several purposes. First, it is used to determine in a qualitative sense the effect of failures of subsystem functions upon the system as a whole. This can potentially flag problem areas where system loss could occur. Second, it requires a first look at the implementation of the health management and fault tolerance for the system as a whole, driving the implementation to the next design level. Third, based upon the proposed techniques, it provides for determination of the time-to-criticality of specific failures. This can be compared to the TTC requirements levied earlier in the design process, determining whether the Fault Prediction, Detection, Isolation, and Response (FPDIR) design can meet the necessary timing constraints to protect the system. The following paragraphs explain the columns of the matrix.

Functional Error: This column contains the symptom of the fault. For example, the loss of the inertial measurement function can be considered to be either large errors which effect control, or small errors which effect navigation, but not control. These cases must be handled differently by the system, and have differing effects. For a power system, an overvoltage condition can either be permanent or transient, and are handled differently (see Figure 3.2-11). If this matrix is completed for the entire system at the subsystem function level (i.e. the power system broken down into power source, conditioning, and distribution, or the control system into sensing, data transfer, data processing, control, and actuation), it provides a substantial aid to the detailed design process, for it identifies the major containment and FDIR mechanisms, timing and data items necessary to perform the preliminary design.

Fault Class: This column contains the fault classification(s) for which the fault belongs to, which can be traced to the initial requirements for how to handle certain fault classes within particular subsystems.

Error Containment: This column contains the location and technique used to prevent the propagation of the symptoms of the fault.

Fault Containment: This column contains the location and technique used to prevent the propagation of the fault itself (so the adjacent components are not damaged).

Detection: This column contains the mechanism used to detect the fault, and how rapidly this mechanism operates.

Isolation: This column contains the mechanism used to determine the location of the fault, isolate it from other possible locations.

Response: This column contains the response to the fault, and how rapidly this response occurs. The response may maintain the system in an operational configuration, or it may put the system into a safe, but non-nominal configuration.

Recovery: This column contains the recovery (if any) necessary to bring the system back to an operational condition, and how long this takes.

Validation: This column contains the method which will be used to validate that the FDIR operates correctly, i.e. fault injection.

Reporting: This column contains the data items or information necessary to report the fault to the system or the system operators.

Variable with Mission?: This column contains a “yes” or a “no” as to whether the fault symptoms or response change according to the mission or system operational mode. If the answer is “yes”, the fault will have to be reanalyzed for each mission mode in which the symptoms or response differ.

3.2.3.2.4. ECR/FCR Architecture

A Fault Containment Region (FCR) is a region in the system beyond which certain Faults (see definition in Section 1.2.1) are not permitted to propagate.

An Error Containment Region (ECR) is a region in the system beyond which certain Errors (see definition in Section 1.2.1) are not permitted to propagate.

The ECR/FCRs can stop just one fault/error or it can block any number of classes of faults/errors. Different regions also exist for the various design levels, i. e.: conceptual, preliminary, and detailed design. In the Conceptual Design Phase, decisions will be made with regard to containment of errors and faults to individual or groups of functional elements.

In the case of an FCR, containing faults to a certain region keeps the failure of one functional element from causing other functional elements to fail. Typically, this protects hardware in one functional area from damaging hardware in another area. **Figure 3.2-12** demonstrates a simple example of an FCR. In this example, the FCR prevents faults in ground commanding from destroying the vehicle.

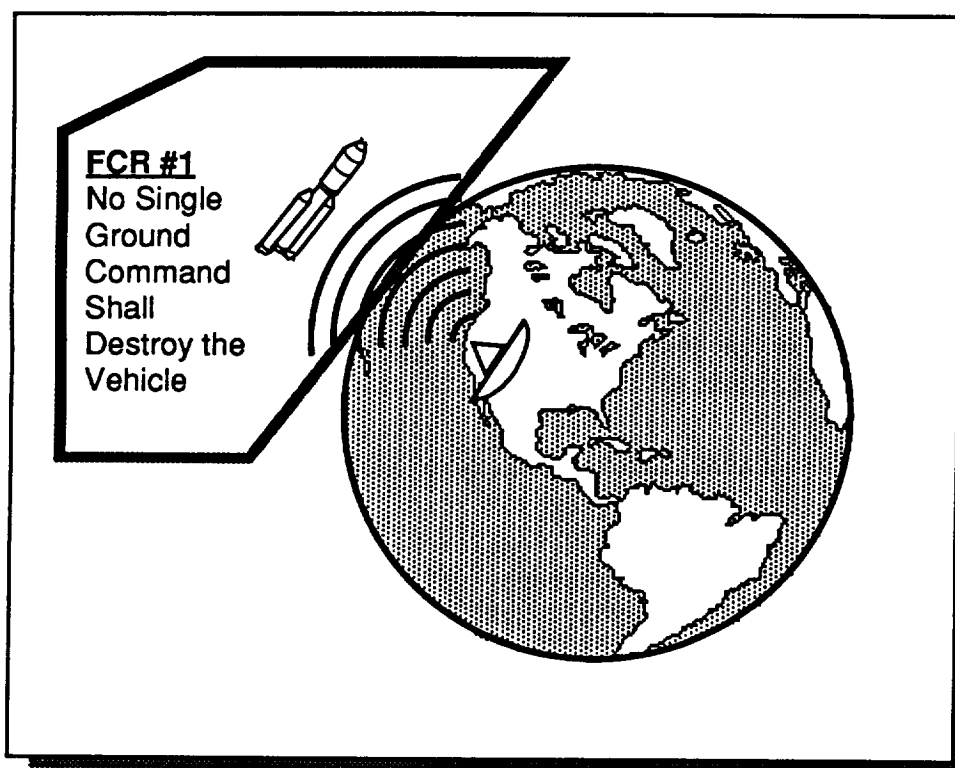


Figure 3.2-12 Example of Fault Containment Region Use

An ECR keeps the symptoms of a fault in one functional element from causing symptoms of faults in other functional elements. These boundaries simplify the fault isolation process, and prevent the effects of the original fault from “contaminating” other subsystems, which would otherwise call for fault detection provisions in the other subsystems. From the standpoint of fault isolation, errors detected within an ECR by definition can not have come from someplace outside the region.

In general, an FCR or ECR can exist at any functional or implementational interface. Thus, the natural evolution of the regions can progress as interface control documents (ICDs) are defined.

It is important to consider entire fault classes when setting containment regions. In certain instances, boundaries which stop certain classes greatly simplify and generalize the design. The generalized design also becomes more robust to potentially unanticipated faults and errors within the class.

As soon as new containment regions are established, they in turn may or may not change the derived requirements for the system. (In some cases, containment regions may be specified in the high-level requirements as determined in the Requirements Phase of the design process.) Once the derived requirements change, a new implementation is established with potentially new interfaces. New interfaces potentially imply new containment regions. Hence, the design of Containment Regions causes an iteration between requirements and implementation. See Figure 2.2-3. When does it stop? The Containment Regions ultimately fall under the auspices of cost, risk and schedule, so a suitable trade takes place for each region and this trade determines the end of the iterative process.

Many methods of enforcing the FCRs and ECRs exist, some of which appear in Figure 3.2-13. However, at the Conceptual Design level, general decisions need to be made as to the approach to implementing each individual ECR/FCR.

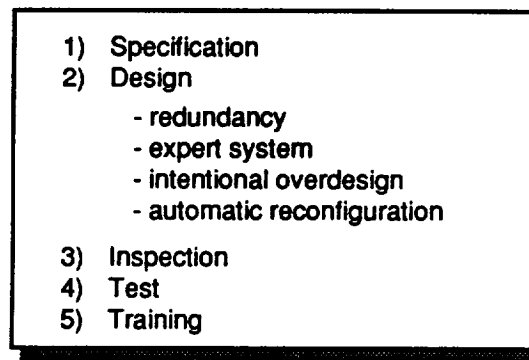
- 
- 1) Specification
 - 2) Design
 - redundancy
 - expert system
 - intentional overdesign
 - automatic reconfiguration
 - 3) Inspection
 - 4) Test
 - 5) Training

Figure 3.2-13. Some Methods of Fault Containment

As an example of enforcing the FCR illustrated in Figure 3.2-12, the designer might choose a multi-layer method: 1) of choosing extensive training for the ground support personnel as the first layer, and 2), then embedding an expert system in the command decoding subsystem.

3.2.3.2.5. Life Usage and Fault Prediction

During SHM conceptual design, it is important to consider system life prediction of when components or systems will fail. (Fault prognostics will be officially added to the fault accommodation list in preliminary design) For reusable systems, the prediction of remaining system/component life is often important if the system utilization time period exceeds the expected life of system elements. Even if none of the flight hardware is reusable, ground systems are predominately in the reusable category. It is important to identify the MTBF and life expectations of the reusable equipment and determine if life usage calculations are desired. These requirements can have a major impact on SHM design. As an example, a vehicle insulation barrier may have a 10 mission life expectation. If active monitoring of the barrier is desired for on-condition maintenance (as opposed to or in addition to between mission ground checks), more on-vehicle parameter measurements are needed. Trade studies as to on-board or ground based inspection options are also conducted. These analyses also consider when to utilize a periodic time or mission usage preventive maintenance plan.

Life usage can be considered the long time phase aspect of prognostics. The short time phase aspect of prognostics is predicting failures before a fault occurs or performance degradation becomes significant. The failure modes identified in the fault accommodation list must be studied to determine whether they are binary in character (good or bad with no in-between), or continuous, with a "gray zones" between good, degraded, and bad. If technology exists to predict an imminent fault before it occurs through the use of trending, pattern recognition, or other techniques, utilization of instrumentation for prognostics should be considered in a cost benefit analysis. Particular attention should be given to the utilization of predictive monitoring and redundant checks where catastrophic effects or critical failures without time to compensate or reliable compensation techniques are available.

3.2.3.2.6. Verification and Validation Plan

The activity of verification and validation generally makes sure that the system, as designed and implemented, consistently performs to requirements and intent. There is inconsistency in the industry regarding the definitions of verification and validation. This discussion will assume the following definitions:

Verification: Ensure that the system, as designed and implemented, performs to requirements.

Validation: Ensure that the system, as designed and implemented, performs to intent.

In the world of health management, verification usually entails demonstrating high levels of required reliability and availability. Validation involves convincing everyone (the designer themselves, management and the customer) that fault mitigation has been worked correctly into the system design and implementation.

One hundred percent verification and/or validation is not possible given finite cost and schedule constraints. See Figure 3.2-14.

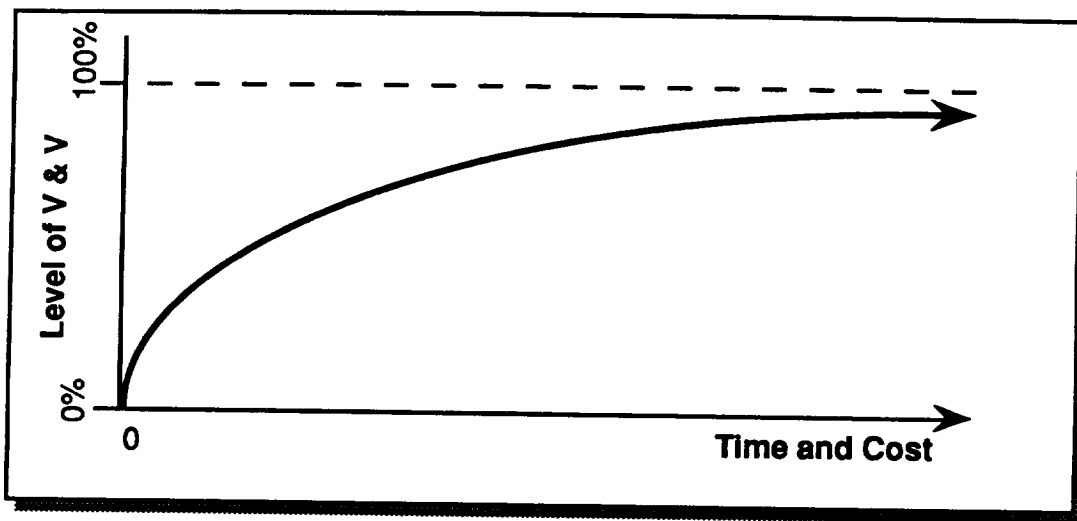


Figure 3.2-14. The Cost of Verification and Validation

When discussing the decision process later on in this section, cost and schedule will tend to lower the level of V & V, whereas the complexity brought on by health management functions (and the fear of the unknown interactions with the rest of the system) will dictate higher levels of V & V. This ongoing trade will determine the level of V & V eventually performed.

In the conceptual design process, the designer always asks whether a given conceptual design meets the criteria of verifiability and validatability. If a design fails to meet those criteria, then the designers must change the design. Hence, the V & V question affects the design process in the earliest phases. The designer can also in some cases design the system in such a way such that faults cause only a limited number of fault symptoms. When very few fault symptoms for a given box or component are manifested at the interface, despite the variety of faults internal to the component, the V&V for that component, and for the surrounding system, from the SHM standpoint becomes much simpler.

In the conceptual design phase, the V & V planner allocates the functional faults identified so far into the appropriate V&V category or categories. This information and expert judgment is used to do a preliminary fault injection analysis. The results of that analysis are added to the functional fault matrix table introduced earlier as Figure 3.2-11. Many ways exist to validate and verify a design. The health management paradigm expands the three classical methods of V & V (analysis, inspection and test) into five: analysis, simulation, subsystem test, system test, and, formal proof. Figure 3.2-15 shows these method's and their levels of fault injection capability, fidelity, degree of certainty and costs.

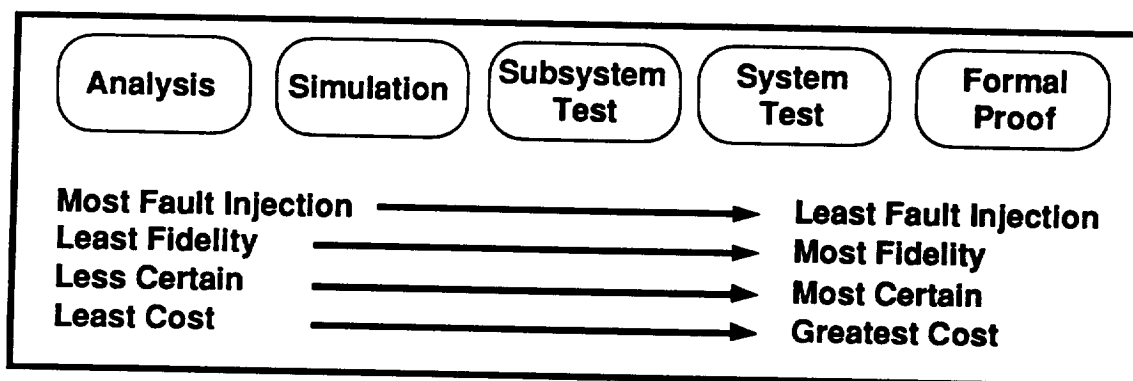


Figure 3.2-15 Methods of Validation and Verification

Analysis generally means collecting analyzing (perhaps by simply thinking about the problem, or performing mathematical or logical analyses of various sorts) and presenting the relevant facts to a V & V question in a convincing manner. From this, it becomes apparent that getting heads to nod in assent doesn't take much money (most of the time), and, genuine reservations can still exist after completing the analysis.

The simulation, subsystem and system test methods comprise the “empirical” methods referenced later in this section. Simulation refers to running tests using computer programs, or other theoretical models. Clearly, this method fails to check faults related to implementation at the lowest level. Hence, those questions always remain unanswered after using this method of V & V. It is, however, a very effective and cost effective method in many cases, for in simulation, virtually any type of fault can be injected into the system, without fear of damaging the system. It thus provides the most flexibility for fault injection, being limited only by the imagination and resources of the programmer.

The subsystem (or unit-level for software) empirical test scrutinizes the actual implementation of a particular subsystem. This gives the V & V planner the next highest level confidence that these kinds of tests answer some of the implementation questions left unanswered by simulation (and maybe vaguely answered by analysis). However, this kind of test seldom investigates the more complex and interesting inter-system, health-management interactions, since the subsystem gets tested all by itself without ever interacting with the rest of the designed system. (The subsystem will only interact with as much as a simulated version of the rest of the system.) The subsystem test is usually able (with good simulated inputs) to simulate many types of failures, with a greater level of reality, although there are some limitations due to the use of real hardware and software.

The system test subjects the system to as much reality as possible and/or practical. (It is often not possible or feasible to run a system test with fuel and oxidizer in the propulsion system or live pyrotechnics on board.) Reality in this case includes system operations in all the anticipated phases of the operational mission and under adverse conditions: thermal, EMI, vibration, shock, turbulence, radiation, pressure, contamination, human mishandling, human operational errors, malfunctions, misalignments, etc. The system test requires the most coordination between the most disciplines, but at the same time does a very effective job at uncovering unexpected complexities in the inter-subsystem health management functions and implementations. The major limitation of this type of testing is that it is often not possible to actually inject various failures into the real system, due to the nature of the design or risk to the system.

The formal proof proves the design of the system based on analytical models and assumptions believed to be true and correct. This proof provides the highest level of fidelity in the V & V world as to correctness of the system, provided the models and assumptions are correct and accurate.

The V & V plan drawn up in the conceptual design phase will consist of an allocation of all the known functional faults to some V & V method and possibly a list of exclusions. Figure 3.2-16 shows how a V & V plan might look.

Functional Faults	Analysis	Simulation	Subsystem Test	System Test	Formal Proof
Fuel Tank Ruptures	X	X			
Separation Failure		X			X
Power Supply Overvoltage				X	
Loss of Guidance Signal	X	X	X	X	
Sensor Data Error	X		X		
Illogical Command Order	X	X		X	
Loss of Space/Time Continuum					X
...					

Figure 3.2-16. An Example V & V Plan

The design of the V & V plan must always seek completion of the actual V & V activity in the least amount of time and at the least cost, yet balance these with the technical necessity of testing the system in the most realistic way possible. From the technical viewpoint, doing all tests at the system level is the ideal. From the cost and schedule viewpoint, all analysis is the best option. The real plan is a balance between these desires. Realistically, it is also necessary to carry several scenarios through analysis, simulation, test, and proof, in order to check the validity of the test mechanisms themselves, the fidelity of different techniques. The following paragraphs discuss various considerations which drive the allocation of V & V to various techniques.

Moderating the cost and time constraints, which tend toward the selection of analysis and simulation for the V & V activity, the added system complexity required by health management considerations pushes the allocation of fault-management-function V & V toward the “high-end” method of system test. In general, the V & V plan should allocate all health management functions that operate across subsystem boundaries to the system test realm. For instance, scenarios involving autonomous and human directed vehicle safing clearly fall into this regime.

The question of health management function's activity during typical mission scenarios also pushes for more system tests. Again, the complexity factor dictates these "full up" tests. Two classic examples demonstrate the need for system tests of this type. First, on the Voyager mission, the Centaur booster delivered the spacecraft into its transfer orbit well within the Centaur's design requirements, but with more spin in the roll axis than the Voyager designers expected. Hence, the spacecraft's health management decided that a problem existed and sent the spacecraft into a safing routine. This routine took about 2.5 hours to complete, during which time, the spacecraft took itself out of communication with the earth where anxious mission operations people were trying to figure out what went wrong and congressmen were wondering why they voted to fund such nonsense. The second example concerns the recent Hubble Space Telescope. During on-orbit checkout, the ground controllers opened the protective door over the main optical mirror. The action of opening the door caused vibration that the attitude control subsystem interpreted as excessive roll, pitch or yaw rates and sent the telescope into a safing routine. The safing routine caused the ground controllers much lost time in reconfiguring the telescope to a normal operational state. (Of course, one of the steps in the reconfiguration procedure concerned testing the mechanism controlling the protective door — the door opened and the telescope went back into the same safing routine again!)

Programs will always start running out of money and pushing deadlines by the time it comes to carry out the V & V plan. If the V & V planner includes a thorough explanation of the rationale used in developing the conceptual-design-phase plan along with the plan itself, then later, the inevitable reprioritization and reallocation of fault scenarios becomes much easier.

Every functional fault that gets allocated in the V & V plan to an empirical method, directly affects the derived requirements for the system simulation and/or test bed. However, and once again, an iterative process takes place between the V & V test plan and the simulation/test bed design: the plan changes the simulation/test bed, which in turn may change the plan.

Besides the requirements on the simulation and/or test bed as discussed in the previous section, the V & V plan also indirectly specifies fault injection methods for the functional faults allocated to the empirical verification or validation realm.

3.2.3.2.7. HMS Conceptual Design

At the end of conceptual design, the products and activities listed in Table 3.2-2 should be completed. The resulting HMS Conceptual Design will be unique to the system being designed. As previously mentioned (Section 3.2.1), the actual form of architecture may differ from subsystem to subsystem but will contain hardware, software, and operational elements. Part of the architecture will include appropriate levels of redundancy, the FDIR scheme, the ECR/FCR definition, the parameters to be monitored, and the degree of system autonomy (ground vs flight based decision partitioning/degree of human role in SHM functions).

3.2.3.3. Preliminary Design HMS Requirements

3.2.3.3.1. Test Bed and Real Time Simulation

Every project seems to defer thinking about the HM details of the test bed and real-time system simulation until just about every other aspect of the system design has been finalized. Thus, come the 11th hour in the design process, someone finally recognizes the boat load of completed HM work required in undoubtedly short notice. Pushing the panic button at this point does get the work done, but at what price! Our methodology emphasizes consideration of as many of the known failures of the system as early as possible. And this applies just as much to the test bed and real-time simulation. Therefore, starting design activities on these tools in the conceptual phase is certainly appropriate. It not only avoids the last-minute design panic mentioned earlier, but it also greatly aids the design and checkout process.

What will go into the simulation and test bed design in the conceptual design phase? Two things will, actually. First, the functional faults identified during the development of the Verification and Validation plan form additional requirements that affect the simulation. Realizing that the system simulator will include several diverse, and possibly remote, processors, coordinating the incorporation of the identified functional faults into the various parts of the overall simulation poses a formidable, but necessary, challenge. Again, better to take on this dragon sooner than later.

Secondly, provisions for the fault injections documented in the functional fault matrix influence the design of the test bed. If for instance, a particular fault injection calls for hitting the tested thing with a hammer, how large of a hammer does the test bed need to get? is there enough room in the test bed to swing the hammer? is a gorilla required to swing the hammer, or can a normal person do it? what protections need to be built into the test bed for the hammer-swinging test? etc., etc. etc.

Aside from the requirements generated by the V & V plan, a number of additional considerations drive the design of the simulation and test bed in the conceptual design phase. First, HM considerations always add to the resource requirements for the system simulation and test bed. Said in another way, don't make simulation and test bed design decisions based solely on the nominal functionality requirements of the system.

Secondly, additional measurements and/or methods of providing visibility into the HM functions need consideration. Providing additional information in the conceptual design phase incarnation of the simulation and test bed will greatly aid the design and check out of the HM (and perhaps nominal) functions. As an example, for any FPDIR activity, time-tagged messages as to prediction, detection, isolation and response makes the difference between night and day in HM design and check out (see **Figure 3.2-17**). Some other examples of visibility measures are: states of isolation valves, active part of a redundant set, out of limit readings, etc.

```

0:45:27.1 CONTROL: inserted failure #12 in function A
0:45:27.3 MODEL: detected failure in function A
0:45:27.3 CONTROL: simulation mode switched to ISOLATION
0:45:27.4 MODEL: isolated failure in function A to #7 or #12
0:45:27.4 CONTROL: simulation mode switched to RESPONSE
0:45:27.4 MODEL: taking fault response #6 for function A
0:45:28.7 MODEL: taking fault response #2 for function A
0:45:29.2 MODEL: taking fault response #3 for function A
0:45:29.3 MODEL: recovered from failure in function A
0:45:29.3 CONTROL: simulation mode switched to NORMAL
0:51:12.0 CONTROL: inserted failure #1 in function C
.....
.....
.....

```

Figure 3.2-17. Time Tagging HM Activities

Thirdly, realize that the test bed and simulation HM needs will expand as the program progresses (see **Figure 3.2-18**). Provide for this expansion in the design of the test bed and simulation in this early phase. Several "for instances" exist. For instance, the subsystem designers develop increasingly detailed simulations of their HM design starting in the conceptual design phase. Wouldn't it be great if they *just happened* to develop these tools such that they plug right into the continually maturing system simulation? The designers would thus use the same programming language as the simulation, the same host processor as the simulation, the same electrical interface as the simulation, the same operational procedures as the simulation.

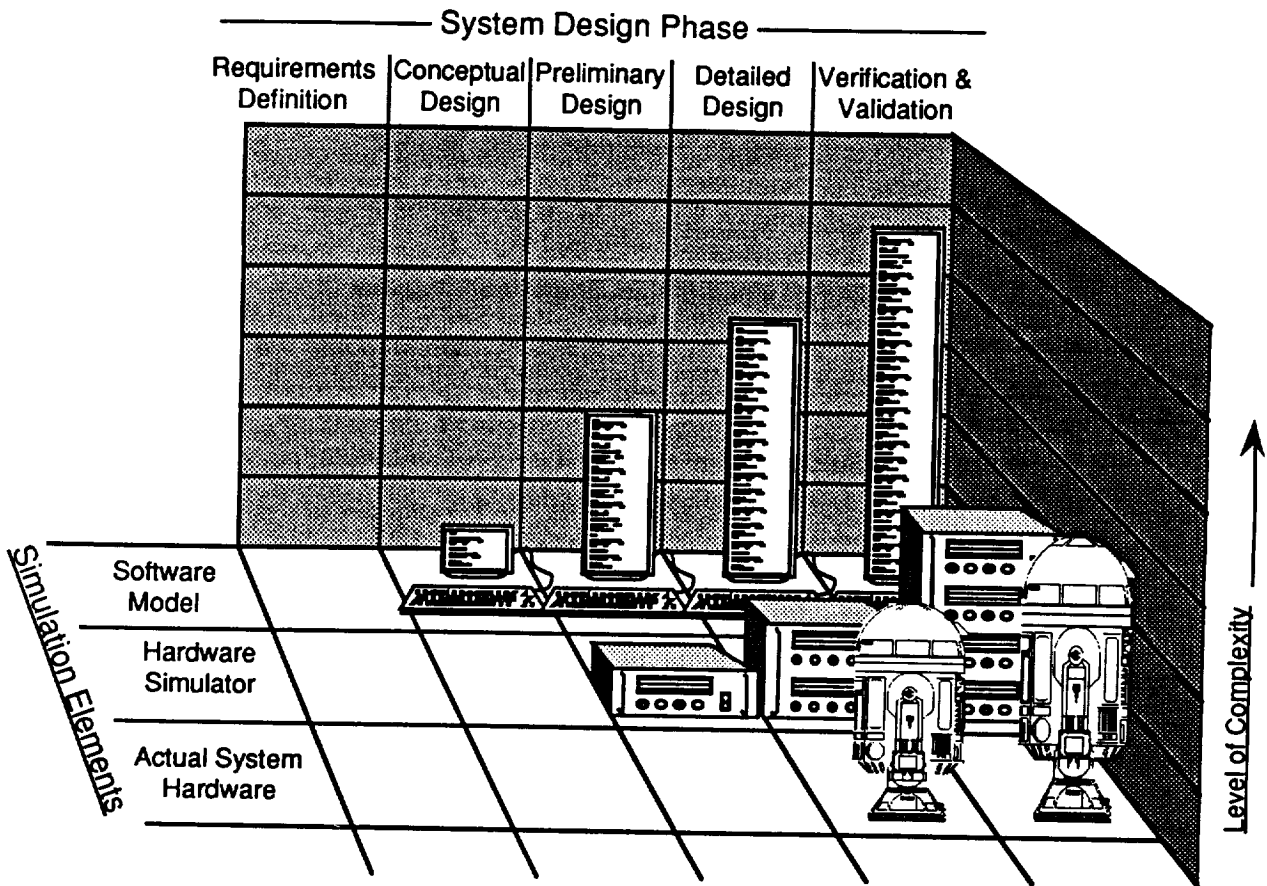


Figure 3.2-18 Provisions for Testbed/Simulation Expansion

For instance, the simulation and test bed matures as concepts become design reality. Where to put the extra stuff required for fault injection? What about the extra monitoring required for fault response visibility? The processing and/or trending of the additional HM information?

Fourthly, the conceptual design phase simulation and test bed design must implement functional faults (and later implementation-level faults) such that they may be injected causally, i.e., within the practicality of the implementation, the capability to inject a given fault in the same way for each injection must exist. This translates into an automated control mechanism for the simulation and/or test bed.

3.2.3.3.2. H/W, S/W, Operations Examples of Requirements Types

Table 3.2-6 shows some examples of Preliminary Design requirements for hardware, software and operations.

Hardware :

- The Engine Controller Interface Shall Be A Fault/Error Containment Region
- The Parameters for Trending On Subsystem X Are: N1, N2, ...
- Their Shall be Redundant Motor Driveson the Electromechanical TVC Actuators
- The Inertial Measurement Unit Shall Be Single Fault Tolerant
- The Data Highway Throughput for Node X Shall Exceed 20 Kblts /sec.

Software :

- Software Retry Recovery Sequences for Error Containment Regions X and Y Shall Execute Without Disrupting Data Storage at Z.
- The Ada Compiler Utilized For The Health Management Routines Shall Be
- Health Management Routines Shall Comply With Testability Specification X.

Operations:

- Operations Trend Data Shall Be Collected From the Following Subsystems: X, Y, Z

Table 3.2-6. Requirements Examples for Preliminary Design

3.2.4. SHM Conceptual Design Phase Summary

Summarized in Table 3.2-7 are the key activities of the conceptual design phase for SHM development.

- List of Faults (FMEA) and Fault Accommodation List
- Functional Fault Matrix (FPDIR Plan, Verification, ECR/FCR., etc.)
- Parameters To Monitor
- Time to Criticality Analysis at Upper Levels
- Establish Subsystem Error & Fault Containment Regions
- Health Management Data Flow Plan
- Early Design Provisions for Verification of System Fault Tolerance
- Degree of Autonomy—Vehicle vs Ground, Degree of Human Role
- SHM Software Plan Integration with Overall Software Plan
- Early Identification of Passive Fault Tolerance Design Provisions
- Develop Requirements for Simulation/Test bed Design
- Identify Requirements for & Begin Development of Analysis Tools for SHM Design Support
- Develop SHM Requirements for Next Phase (Preliminary SHM Design)

Table 3.2-7. Key activities for Conceptual Design Phase

3.3. Preliminary Design Phase

In the SHM preliminary design phase, the health management system at both the subsystems and systems level becomes much better defined. As rough concepts for detecting faults, containing faults and errors, and making provisions for accommodating the less publicized types of faults (such as intermittents, Byzantine, and human errors) are translated into design solutions, the HMS implementation takes serious steps toward becoming reality. Recognizing that the degree of technical difficulty and the payoff of creativity rises significantly during this design phase, the SHM methodology provides a systematic design approach guideline, but still relies upon subsystem and component design engineers to generate innovative ideas to purposely deal with faults.

HMS concepts are again iteratively refined and trade studies conducted to investigate alternative methods of meeting the system dependability requirements. And, similar to the other design phases, a great deal of iteration takes place between activities within the phase as well as between this phase and other design phases.

As a case in point, the level of detail of the vehicle's system design is substantially better defined during this phase, and many conceptual design ideas can require substantial modification or revision for either technical or schedule reasons. For example, it is often discovered that a desired design concept or sensor is too technically immature to commit to by the design freeze date. Or again, a conceptual design architecture chosen in conceptual design may be unable to handle the data throughput level that is discovered upon better definition and subsequent expansion of a sensor input list. Thus, iterations between concept and design take place.

Figure 3.3-1 diagrams the flow of activities in this design process. The shaded rectangle depicts the design process with the intersecting circles representing that portion of the system design where the dealing with faults impacts the functionality of the nominal system design.

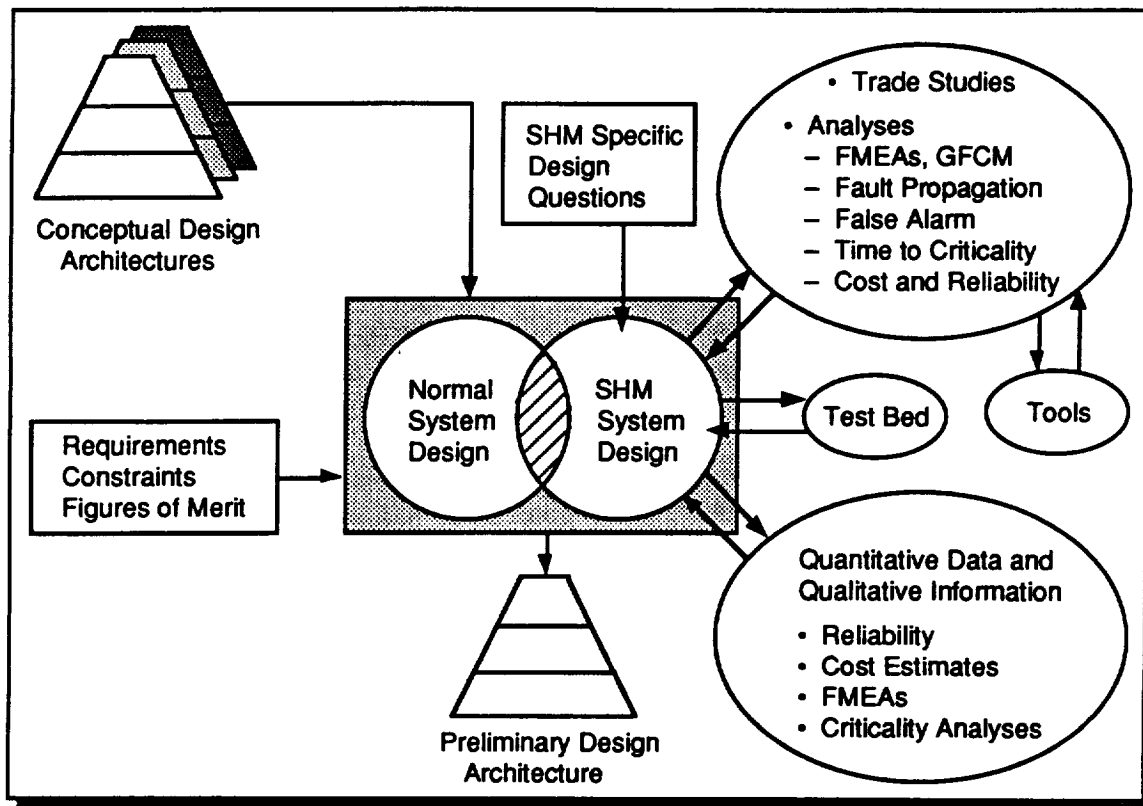


Figure 3.3-1. SHM Preliminary Design Overview

A number of things feed into the overall system design. The first is the conceptual design architectures that originated in the conceptual design phase. The next are the ever-present requirements, constraints and figures of merit. The rest of the diagram deals with specifically the SHM design: the SHM specific design questions, the trade studies, the test bed and the various pieces of information and data. And, as indicated, trade studies influence the SHM design which in turn influence the trade studies. The change in trade studies creates the need for a different tool. The test bed and the SHM design interact in a similar way. Also, new information discovered also changes the design which in turn causes new data and information.

With regard to the design-area overlaps in the figure above, it is difficult to segregate many design decisions as SHM or general systems engineering decisions. This becomes especially apparent in the preliminary design phase because many hardware and software elements address both health management and other basic functions. Examples of decisions that are both systems engineering and health management related are shown in Figure 3.3-2.

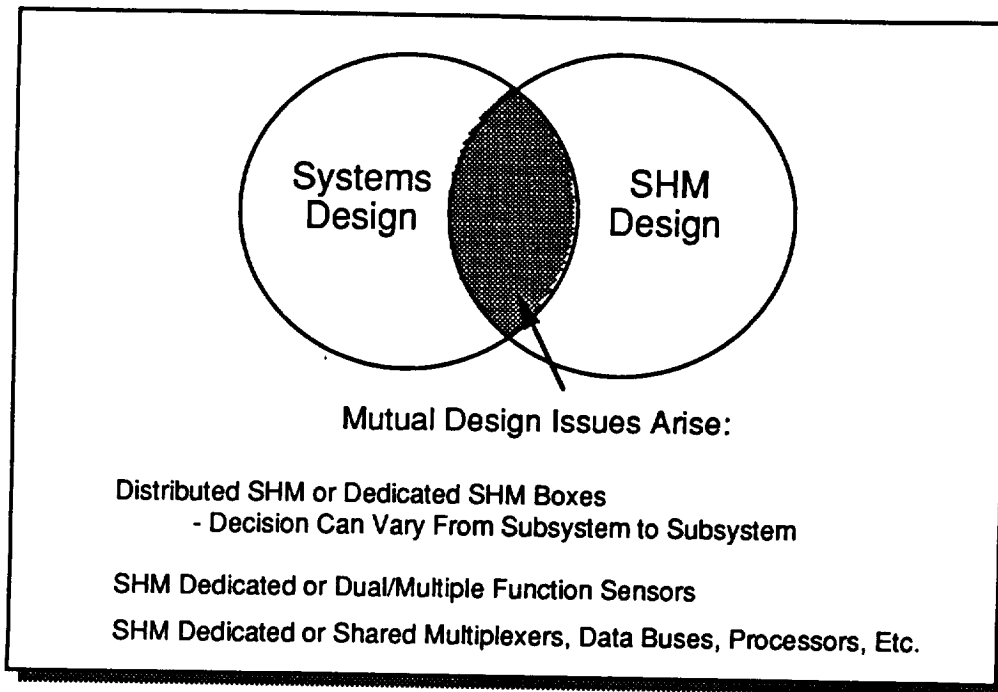


Figure 3.3-2. Mutual Systems Design and SHM Design Decisions

3.3.1. Preliminary Design Phase Objective

The objective of this phase is to take the health management system concepts developed in the conceptual design phase (indicated in the upper left corner of the **Figure 3.3-1**) and downselect to a preliminary health management system architecture like that indicated at the bottom of the figure. In order to do this, this phase develops a definition of the health management system at the subsystem level remembering (keeping the blinders off, so to speak) that the SHM functions will possibly interact with other subsystems.

More specifically, at the beginning of this phase, the fault accommodation list developed during the conceptual design phase consists of a rough list of fault classes and fault handling/prevention provisions to the best level known for each subsystem. (Some subsystem SHM designs will have progressed to the parameter identification level where those subsystems are already well defined.) The goal of this design phase is to come up with parameter lists, algorithm approaches, and corresponding sensors identified for all subsystems and the integrated SHM network.

3.3.2. Preliminary Design Phase Major Activities

The activities and products of the preliminary design phase are shown in **Table 3.3-1**. These activities and products correspond to the four bubbles on the right hand side of **Figure 3.3-1**.

- Complete Fault Prediction, Detection, Isolation, and Response (FPDIR) Plans Hardware, Software, and Algorithmic Approach
 - Levels of Redundancy
 - Degree of Cross Strapping
 - Utilization of Voting, Hot or Cold Standby, Particulars of Redundancy Implementation
 - Threshold and Persistence Levels for Alarms
 - Refinement of Parameter List, Combining Parameters With Data Fusion for Information Confidence
- FPDIR Design Analyses:
 - Detailed FMEA and Augmentation of FMEA With:
 - Quantitative Failure Rate Estimates
 - Gathered Fault Combination Analysis
 - Fault Propagation Analyses
 - False Alarm Analyses
 - Time to Criticality (Using Fault Propagation Analyses)
 - Cost Effectiveness and Reliability Analyses
- Preliminary Sensor Selection for Parameters
- Design Specifics for Layered Error/Fault Containment Regions (ECR/FCR)
 - Hierarchy of Service and Control
 - Partitioning and Allocation of Functions and State Variable Sets
 - Communications Protocol
 - Hardware and Software Partitioning, Fault Classes for Partitioning
 - Isolation Requirements and Mechanisms
- Refinement of Architecture Concepts at System and Subsystem Level
 - Network Nodal Arrangement, Protocol, and Timing
 - Modes of Operation of Each Subsystem and Mode Compatibility
 - Signal Conditioning, Multiplexing, Data Processing, Data Storage/Retrieval
 - Identification of Requirements for Control System Performance In Off-Nominal Fault Induced State
- Provisions for Retest/Recovery/Reintegration for Switched Out Components and Subsystems
- For Fault Responses of Design Out, Inspect and Checkout; and Detect and Report Only, Identification of Personnel, Equipment, Training, Procedures, and Other Means To Implement These Procedures
 - Quantified Requirements for Inspection and Test
 - Human Interface Requirements
- Test Bed/Simulation Setup
- Refinement of Health Management Data Plan
- Special Provisions for Ground System Health Management
- Requirements Refinement for Next Design Phase

Table 3.3-1. SHM Preliminary Design Activities and Products

FPDIR

During preliminary design, a major emphasis is placed on fault prediction, detection, isolation, and response (FPDIR). For this, parameter measurements, sensors, and associated algorithms are required. Algorithmic implementation options include rule based or model based reasoning (such as banks of Kalman filters, redlines, neural networks) and numerous other techniques.

FPDIR Analyses

Several analyses serve to support the FPDIR plan.

- Failure modes and effects analyses are conducted at the system and subsystem level. It is crucial from an SHM design perspective to apply the FMEA toward all fault classes specified by the SHM requirements work. Interaction faults must also be addressed. FMEA data must be augmented by quantitative failure rate estimates. Another FMEA follow on is a gathered fault combination method (GFCM) analysis. In GFCM analysis, faults are gathered according to effect and consequences, with a heavy emphasis on system interaction type faults.
- Fault Propagation Analyses: Faults are postulated, and analyses conducted as to the propagation sequences of the faults to either fault containment region boundaries or fault tree “top event” consequences. These analyses encompass both functional and hardware/software domains, and are very closely coupled with time to criticality analyses.
- False Alarm Analyses: For each detection that has a decision pathway associated with it, a false alarm analysis should be conducted. In some cases, false alarm consequences can be devastating if actions are initiated without a high confidence and isolation of an anomaly.
- Time to Criticality Analyses: The substantially better subsystem definitions available during preliminary design enable both functional and subsystem time to criticality analysis time constants to be much better defined. Time to criticality is broken into detection, isolation, response determination, and response effectiveness segments during preliminary design.
- Cost and Reliability Analyses: Correlation of specific design features to cost and reliability is conducted using a variety of tools. Central to these analyses is understanding the uncertainty levels associated with the cost and reliability results.

Sensor Selection and Parameters

The sensors used in the system as well as the parameters they measure or sense will be chosen based on several trades and analyses. These include, but are not limited to, built in test, stability over time, accuracy, etc.

Partitioning the System Into ECRs FCRs

Drawing boundaries in the system over which certain classes of, or individual instances of, faults and errors will not be permitted to pass constitutes the most organized and comprehensive way of designing the SHM portion of the system. Wherever interfaces exist, decisions as to what kind of boundary will exist will be made.

Other Major Activities

The rest of the major activities shown in the table above are by no means unimportant, but they will be discussed in greater detail later on.

3.3.3. Preliminary Design Approach

3.3.3.1. Inputs to Preliminary Design Process

Preliminary design begins with the conceptual health management architectures. Two major tables that partially define these HMS architectures are the fault accommodation list and the functional fault matrix. These tables, plus the other activities and products listed in conceptual design Table 3.2-2 define the SHM architecture status at the beginning of preliminary design.

The figures of merit and constraints from conceptual design, developed in coordination with the entire vehicle and ground system design team, will be well defined at the start of the preliminary design. Life cycle cost will undoubtedly be a major figure of merit for most designs. Likewise, the integrated project schedule will define many time constraints. All of the preliminary design trades will be driven by cost and schedule constraints.

3.3.3.2. Trade Studies

HMS implementations always involve trading performance for dependability. The effects of these trades are felt most strongly in this phase of the design process. The paragraphs that follow by no means discuss all potential trade studies, but at least touch on the important parts of parameters, sensors and effectors — the means whereby the system detects and interacts with the external world.

3.3.3.2.1. General Parameter Selection Criteria

Parameter selection is an extremely crucial part of HMS design. The parameters must be chosen such that:

- a) Fault information content is maximized for the HMS design expenditure
- b) Data is reliable

- c) The parameters measured truly characterize the phenomena/fault
- d) False alarms and failure to detect are rare
- e) Parameters selected have associated sensors that are reliable, affordable, maintainable, and are design compatible. By design compatible, it is meant they do not degrade (or significantly degrade) the reliability, producibility, cost, and general quality attributes of the host component/subsystem.
- f) The parameters selected are cost effective.

3.3.3.2.2. Non-Economic Justifications for Parameters

Safety and control are two significant drivers for parameter justification. Many parameters are justified solely for these reasons. Parameters of significant enough value for use in control and safety functions are often also of great value for trending, health indication, and other health management uses. Sometimes the health management use is a second usage elected because of the availability of the parameter. Costing is complicated in these instances where multiple purposes drive the selection.

Health management use of a control parameter may have an impact on the cost of the associated sensor(s). If health management usage adds requirements to the control sensor such as a need for better absolute accuracy and long term stability for trending, the health management function may not be “free”.

3.3.3.2.3. Parameter Characteristics

Binary and Continuous Parameters

Some parameters are binary (two value), and some continuous. Decisions are generally easier to make from binary parameters than continuous types. Digital systems abound in binary information such as parity bit check good or bad. Exact comparison checks of data buffers are likewise “true” or “false”. With continuous parameters, setting absolute levels is often challenging. How hot is too hot? Is 183.3 degrees acceptable and 183.4 degrees over the limit?

Parameter Time/Repetition Check

To reduce the occurrence of false alarms, many parameters are subjected to time or repetition requirements before being passed through an error/fault containment barrier. As an example, a parameter may be required to remain over limit for two-three computer cycles before an alarm is triggered. Or, upon failure or agreement of a parity bit check, the check is repeated and must be duplicated a second time before a decision is made or the data passed to another buffer.

Reasonableness Check

Continuous parameters are frequently subjected to reasonableness checks. These checks determine if the signal is within both the calibrated range of the instrument and realm of feasibility for the subsystem.

Simple, Data Fused, and Synthesized Parameters

A simple parameter is one formed from the input of one or multiple sensors all measuring the same physical quantity, property, or condition at a common location with a common transduction principle. As an example, a simple parameter could be formed by averaging three thermocouples at the turbopump inlet plane. A data fused parameter is usually thought of as one fusing data from different locations of measurement (spatial), different times of data collection (temporal), and/or different types of transducers. As an example, a turbopump inlet temperature fused parameter could be formed from the average of a resistance temperature detector and a thermocouple measuring turbopump inlet temperature. Or, a propellant density fused parameter could be calculated from a temperature and pressure measurement, via a lookup table or formula.

A synthesized parameter is one calculated from other related measurements and/or a mathematical model of the process. Aerothermodynamic relationships support computation of flow path variables at selected locations from parameters elsewhere. When model based reasoning is utilized in HMS systems, extensive parameter synthesis is possible. In the absence of a complete model of the process, known empirical correlations between parameters enables the synthesis of parameters.

3.3.3.2.4. Fault Accommodation Cost Effectiveness Trades

Fault accommodation involves performing trades to decide upon the degree of design out, test and inspect out, fault tolerance, and report only measures to accommodate the fault identified. Accommodation provisions can “buy their way in”, or be mandated into the design process by qualitative constraints and decisions. Of the four categories listed, design out and fault tolerance are the most friendly for cost effectiveness analysis. Test and inspect out parameter justification is quantifiable, but often difficult. (Built-in-test is usually designed to qualitative requirements such as faults all boards shall indicate functionality). Cost effectiveness analysis of selection of individual parameters for post flight accident investigation is extremely difficult.

Design Out: Design out is always utilized to eliminate faults to some degree. Utilizing better margins, wise material selections, lower parts counts, simplicity, and a myriad of lessons learned, the design engineers conduct trade studies to trade cost and fault avoidance.

Test and Inspect Out: Preventative measures to detect faults after component manufacture include non-destructive evaluation, built-in-test, and assembly/integration checkouts with external equipment. Cost versus effectiveness of these inspections, checkouts, and built-in-tests must be evaluated for the various faults.

Fault Tolerance: Both passive and active fault tolerance are utilized. For active fault tolerance, parameters to detect and isolate the fault are identified, and then evaluated for their cost effectiveness in enabling the system to tolerate the fault.

Detect and Report Only: For many non-critical faults, or critical faults with low probability of occurrence or lack of economical methods to tolerate, detect and report only is sometimes selected. This data is the primary information to determine the cause of flight incidents or failures. By reducing the ambiguity zone on fault cause, millions of dollars of post accident investigation time can often be saved. Translating specific parameters into dollars saved for post flight accident investigations is an area requiring more investigation.

3.3.3.2.4.1. Flight Fault Tolerance Parameter Selection (Flight Failure Prevention Cost Effectiveness Analysis)

Flight fault tolerance parameter selection involves deciding on what phenomena are to be measured to detect and prevent flight loss through active fault mitigation. For launch vehicles, cost benefit studies consistently show that the greatest economic benefits arise from flight loss prevention. Faults are detected and isolated by deciding upon a symptom set that hopefully uniquely corresponds to a particular fault. Each fault detection selected must have a corresponding parameter or parameter set upon which the detection is triggered or set to "true". Selecting the most economical and reliable parameters for a desired detection is a key part of HMS design. Generally, the most direct measurement of the desired detection is favored over indirect inference from secondary parameters.

The predicted frequency of occurrence of faults can be found in Table 3.2-5. As shown in Figure 3.3-3, flight failure rate is converted into a compensable failure rate, which is then converted into an effective flight failure reduction number, often in units of parts per million for convenience. The effective flight failure reduction number is converted into dollars through use of a cost model which accounts for the cost of a lost mission, including downtime.

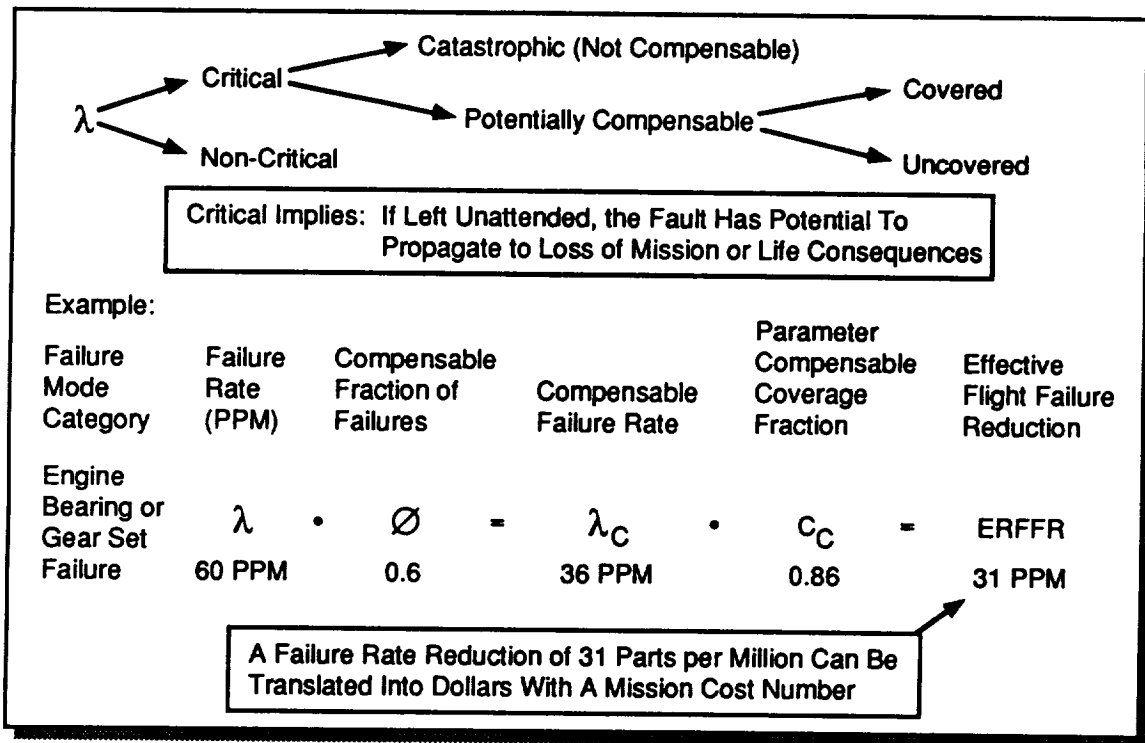


Figure 3.3-3. Failure Criticality and Compensable Failure Coverage

The compensable coverage fraction depends upon the choice of parameter or parameters to detect and isolate the failure. Time to criticality analyses support these parameter choice trades by determining the time frame within which the detection and isolation must function. The cost and estimated effectiveness of different parameters or parameter sets to address each fault are then tallied, and compared against the potential flight loss prevention cost savings. An example of the savings, costs, and a net benefit of a parameter/parameter set or flight loss prevention are shown in the bar chart formatted **Figure 3.3-4**. Saving attributable to other factors besides flight loss prevention are also shown in the figure and will be discussed next.

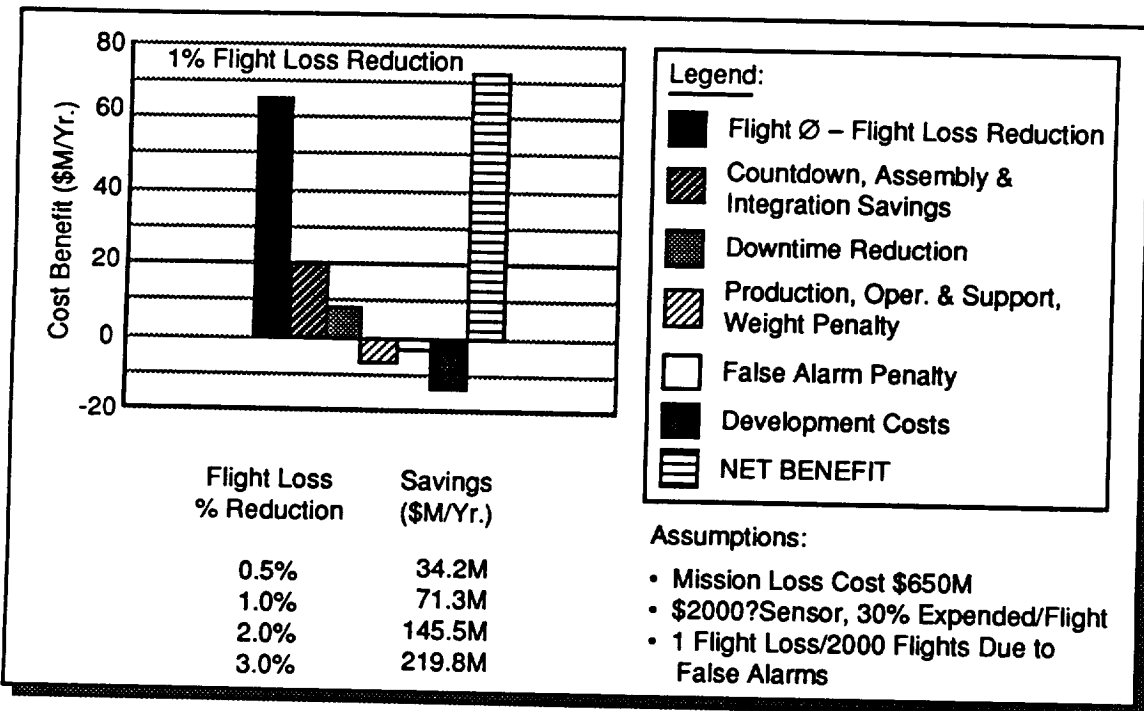


Figure 3.3-4. Typical Parameter Economic Effectiveness Assessment

Due to the sparsity of quantitative failure data for some subsystems, a high, low, and best estimate (average) frequency of occurrence of failures might be required. The economic analysis should be run for each estimate. If a parameter is economically justified for the average expected failure rate, the parameter or parameter set should be considered to have bought its way into the design. Some design groups believe that a parameter justified on the basis of a worst case failure rate estimate should be considered "bought in". This is an issue that should be elevated to the project management and customer level.

3.3.3.2.4.2. Cost Effectiveness Based on Factors Other Than Flight Loss Prevention

As suggested in Figure 3.3-4, although flight loss prevention is typically the “heavy hitter”, other factors can economically justify parameters in addition to flight loss prevention. Shown in Table 3.3-2 are typical cost saving contributions from health management parameters by use.

<u>Typical Economic Significance of Contribution</u>	<u>Parameter Justification</u>
High	Flight Loss Prevention
Moderate	Startup/Shutdown Monitoring (Part of Flight Loss Prevention)
Low	Assembly/Integration Checkout Use
Low/Moderate	Expedite/Assist Post Flight Incident or Failure Investigation
Low	Life Usage Indicator/ Condition. Mon. for Maintenance
Justified by Control Use	Dual Use as Control/Health Management Parameter
Justified by Safety Usage	Dual Use As Critical Safety Parameter

Table 3.3-2. Cost Savings Contributions From Parameters By Use

Flight loss prevention has been extensively discussed. Other justifications are:

- 1) Benefit for system start up and shut down protection. This can be considered part of flight loss prevention. A better snapshot of the engine startup can improve ascent reliability if holdown is utilized. The frequency of failures during startup can be translated into flight loss savings in a cost model.
- 2) Benefit for Post Flight Incident/Failure Reporting. If a flight loss occurs, more extensive data enables the accident investigation to proceed more confidently and expeditiously. This translates into shorter downtimes after accidents, and huge savings as a result. The challenge is determining how much each added parameter contributes to downtime reduction.

3) Assembly/Integration Checkout Improvement Added parameters can provide a higher level of confidence of system readiness during vehicle checkout and assembly. In turn, system checkout can move faster where additional well organized information is available. Parameters are just one of many factors contributing to checkout and assembly however. It is very easy to have additional parameters, "paperless" management, a more efficient assembly floor layout, better database availability, etc. take credit for the same improvement. Double accounting can very easily occur, especially if each group is trying to justify its particular technical contribution.

4) Life Usage/On-condition Monitoring: This can be considered a subset of assembly/integration checkout improvement. For reusable equipment / systems, parameters measured to compute the life usage or the time when maintenance is needed can be useful. For most concepts, this justification for parameters is more applicable to ground systems.

3.3.3.2.4.3 Complexities

The economic analysis varies in complexity from subsystem to subsystem. The complexity of the economic cost effectiveness analysis for parameter selection is effected by:

1) Uncertainty of failure rate data: To address this, a high/average/low approach is suggested coupled with an estimate of failure rate uncertainty for each subsystem.

2) Failure propagation and the effectiveness of parameters to detect failures several tiers away in a failure propagation diagram: shown in **Figure 3.3-5** is an engine component functional schematic for the Aerojet first stage engine on the Titan launch vehicle.

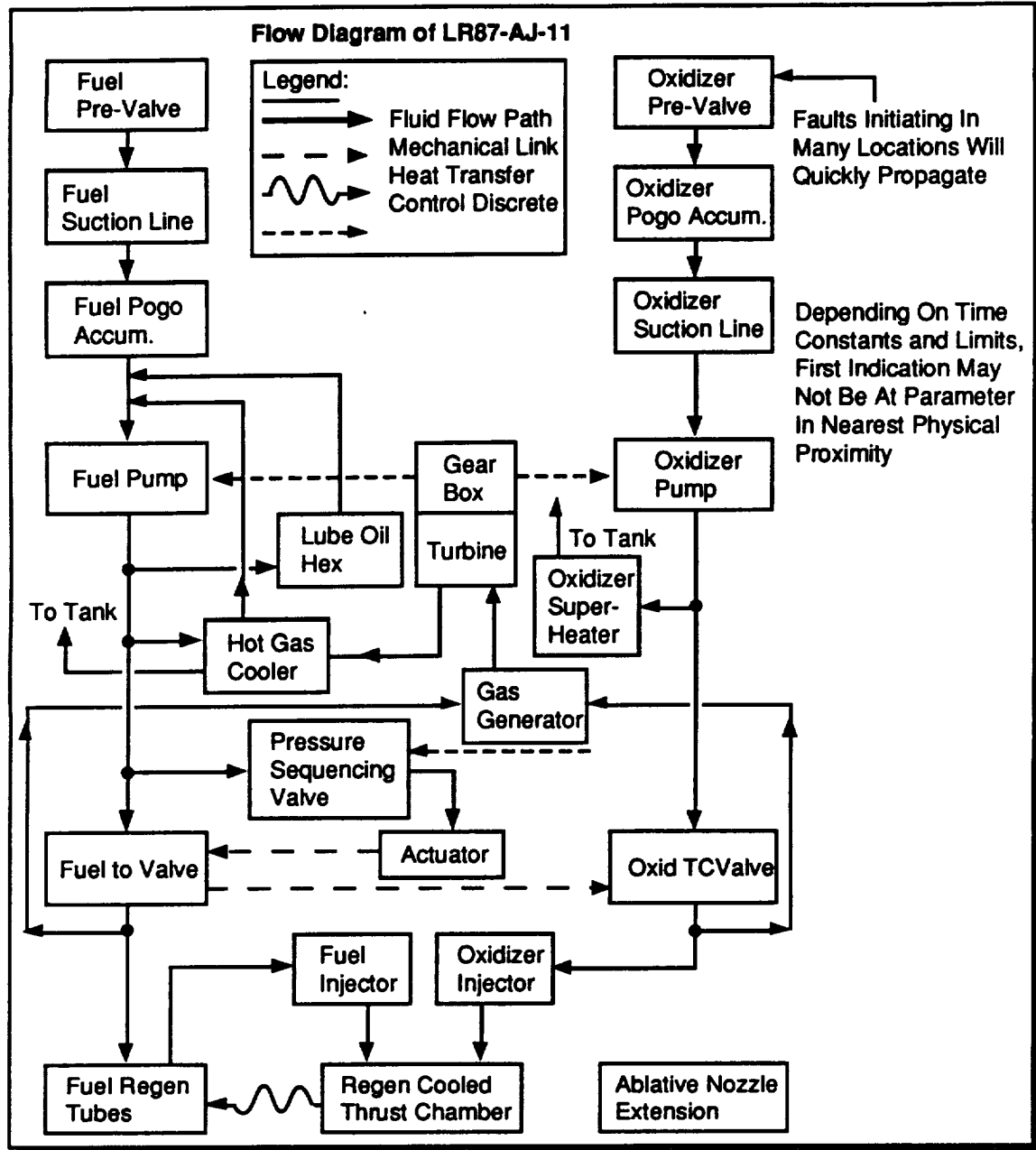


Figure 3.3-5. System Coupling Complicates Fault Isolation

A failure at the oxidizer pre-valve which severely effects the oxygen flow will be detected by a pressure transducer on the oxidizer suction line, but also effect numerous other parameters such as engine main combustion chamber pressure. Using time to criticality analysis, if main combustion chamber pressure can detect the oxidizer pre-valve failure in time for a successful shutdown, this parameter can be used in lieu of oxidizer suction line pressure. From an analysts standpoint, care must be taken to avoid either double credit being taken by a parameter for the same failure, or credit not being attributed to parameters one or more tiers removed that have significant visibility into a failure.

3) Cost Benefit Bookkeeping if Several Parameters Share the Detection of A Failure: If a parameter makes a contribution to detection and isolation of several failures, assignment of fractional credit can get complicated. An approach to this situation can be a cost benefit analysis conducted on a small parameter set, where the set can be justified, but the precise individual contributions remain murky.

3.3.3.2.5. Sensor Selection

Parameter selection and sensor selection cannot be decoupled. Whenever a parameter is selected, an understanding of the technical and schedule feasibility of the complete front to back metrology implementation of that parameter must be kept in mind. This includes the sensor, signal conditioning, processors, software, algorithm approach, data storage, cabling, etc. If a parameter requires an extremely costly or yet undeveloped / unproven sensor type, the parameter may have to be eliminated from contention. If a chosen sensor requires an intrusive installation that significantly effects the inherent reliability of the host element, the net benefit could be negative. Alternatively, if non-intrusive installation is utilized, or the sensor is installed such that no significant reliability impact on the host occurs, a net benefit occurs. With good engineering practice, intrusive sensors can be added to many systems without a significant reliability effect to the host, or a benefit gained to risk added ratio of several orders of magnitude. Quantitative estimates should rule over emotion. There is no substitute for designer experience in sensor selection and installation.

Sensor Development and Date of Availability

The development schedule of the launch vehicle plays a major role in sensor selection. The design "freeze" dates must influence how much new sensor development is elected. Backup choices and alternatives should be planned where higher risks and availability doubts occur.

Sensor Requirements

Many textbooks have been written on instrumentation development and sensor selection processes. Some of the key technical criteria for sensor selection are shown in **Table 3.3-3**.

- Absolute Accuracy (Static and Dynamic)
- Time Constant for Response
- Accuracy Dependence on Environment
- Resolution
- Repeatability
- Short and Long Term Stability
- Environmental Compatibility (Temp., Vibration, Pressure, Cycles, Etc.)
- Sampling Rate
- Output Rate
- Availability
- Maintenance Characteristics
- Spares Availability
- Special Attention to Accelerometer Signal Conditioning
- Other

Table 3.3-3. Typical Sensor Requirements

Where sensors have dual utilization for both control and health management, a careful requirements comparison is needed. Control utilization often requires a critical data bus, whereas some of the less critical health management utilizations do not. For long-term trending of ground system or multi-use hardware, health management may require more-long term stability and accuracy than that required for control.

The degree of built in test (BIT) sophistication and testability of sensors and their associated circuitry should be adequately addressed by the requirements.

3.3.3.3. Preliminary Design Analyses

3.3.3.3.1. FMEA and the Gathered Fault Combination Method (GCFM)

The FMEA usually only identifies the single failures. Since interactions and combinatorial causes of faults are so important, it is recommended that the FMEA and associated fault accommodation list be augmented by the gathered fault combination method (GCFM) to better expand the fault domain to include interaction and combination type faults.

Chapter 11 of [Villemeur 91] details the gathered fault combination method (GFCM) of studying failure combinations and interactions which produce undesirable events.

The basic steps involved in the gathered fault combination method are shown in Table 3.3-4.

1. FMEA for Subsystems – Input
2. Identification of “Internal Gathered Failures”
 - Grouping of Failure Modes Which Either Alone, or Combined With Other Failures In Any Combination, Produce the Same Effects or Consequences
3. Identification of “External Gathered Faults”
 - Same As Above, Except the Faults External To a Subsystem Which Produce the Same Effects or Consequences Are Identified
4. Creation of “Global Gathered Faults”
 - Both Internal and External Fault Combinations To Produce the Same Effect/Consequences Are Collected

Table 3.3-4. Gathered Fault Combination Method (GFCM) Steps

“Gathering” is collecting faults together which either alone or combined with other failures produce the same effects independent of combination. Global gathered faults are then collected, consisting of combinations of internal and/or external faults which produce the same effects on the subsystem or system studied.

To simplify the GCFM analysis, only failure modes with an occurrence probability above a predetermined value can be carried forward.

FMEA Augmentations

At this point, at least three additions/augmentations to the FMEA must be made. These are:

- a) Expansion of FMEA to include full set of applicable fault classes.
- b) Addition of failure rate estimates to FMEA faults. Identification of uncertainty bands of failure rate data.
- c) Utilization of FMEA data for gathered fault combination analysis.

Experience - Process and Human Fault Importance

Aerospace experience in a wide spectrum of launch vehicle and space vehicle programs has shown that human induced faults and process failures account for a significant percentage of the overall fault set. It is therefore imperative that the FMEA and GCFM analyses include these factors.

Critical Faults and the FMEA

As part of the FMEA analysis, the analyst makes estimates as to the impact of a particular fault on the system as a whole. The categorization process is shown in Figure 3.3-6.

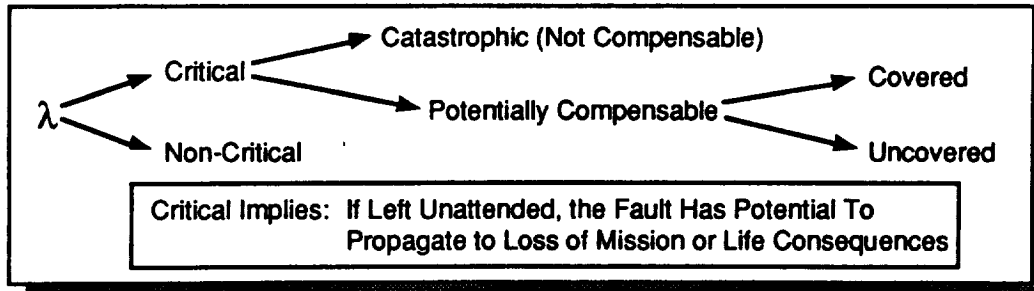


Figure 3.3-6. Failure Criticality Categorization Process

A critical fault is one if which left unattended, has the potential to propagate and cause loss of mission. (For man-rated systems, a revised definition of critical is needed, for the economic analysis becomes secondary to safety mandates). Critical failures are then categorized as catastrophic or potentially compensable. The catastrophic failures are those that are non-compensable by virtue of too rapid occurrence for compensation (extremely short time to criticality) such as a sudden detonation, or, failures for which there is absence of a compensating action such as rapid failure of a critical primary structural element. If monitoring and compensation are to economically justify themselves from a flight loss prevention perspective, it must occur by addressing the potentially compensable failure category.

3.3.3.3.2. Fault and Fault Propagation Modeling

When a fault occurs, the rate and extent of propagation of the effects of the fault outward from the source is very design dependent. The design of layered error and fault containment regions is heavily dependent upon an analysis of the rate and pathways of fault propagation. Figure 3.3-7 shows an initiating event propagation analysis.

Fault Propagation Sequences Must Be Analyzed for Time to Criticality

- Parameter Response Time Requirements Are Determined As Well As Associated Sensor Time Constants

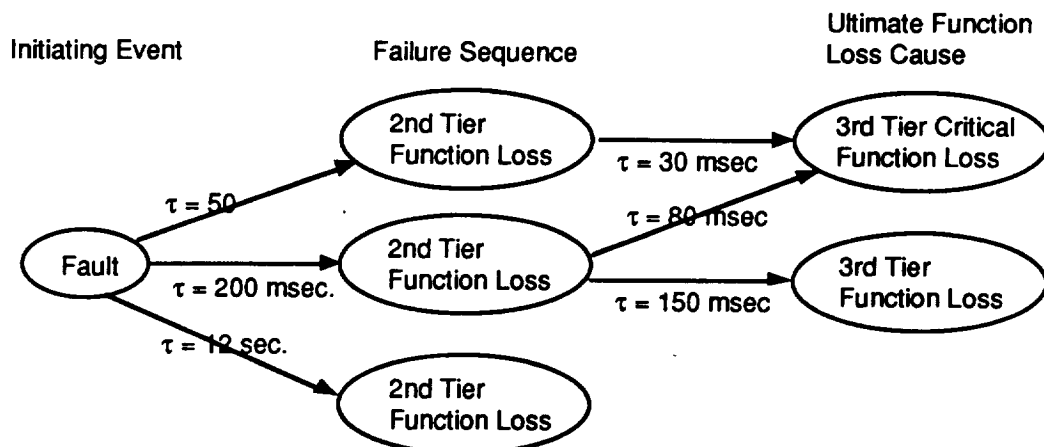


Figure 3.3-7. Fault Propagation Time to Criticality

Key to the analysis is the dynamics of the fault propagation, i.e., how fast and through what pathways does the fault propagate? For systems with few fault containment regions, the cascading of effects can be severe, and the complexity of the analysis very difficult, especially if non-linearities, time lags, and state dependent factors are present. By state dependent factors, it is meant propagation dependency on the operating mode or point in the cyclic/non-cyclic varying condition of elements in the propagation path.

In the chemical/nuclear industry, a fault propagation analysis (without time constants) is known as a cause-consequence diagram (see **Figure 3.3-8** for a cause consequence diagram for an avionics cooling system).

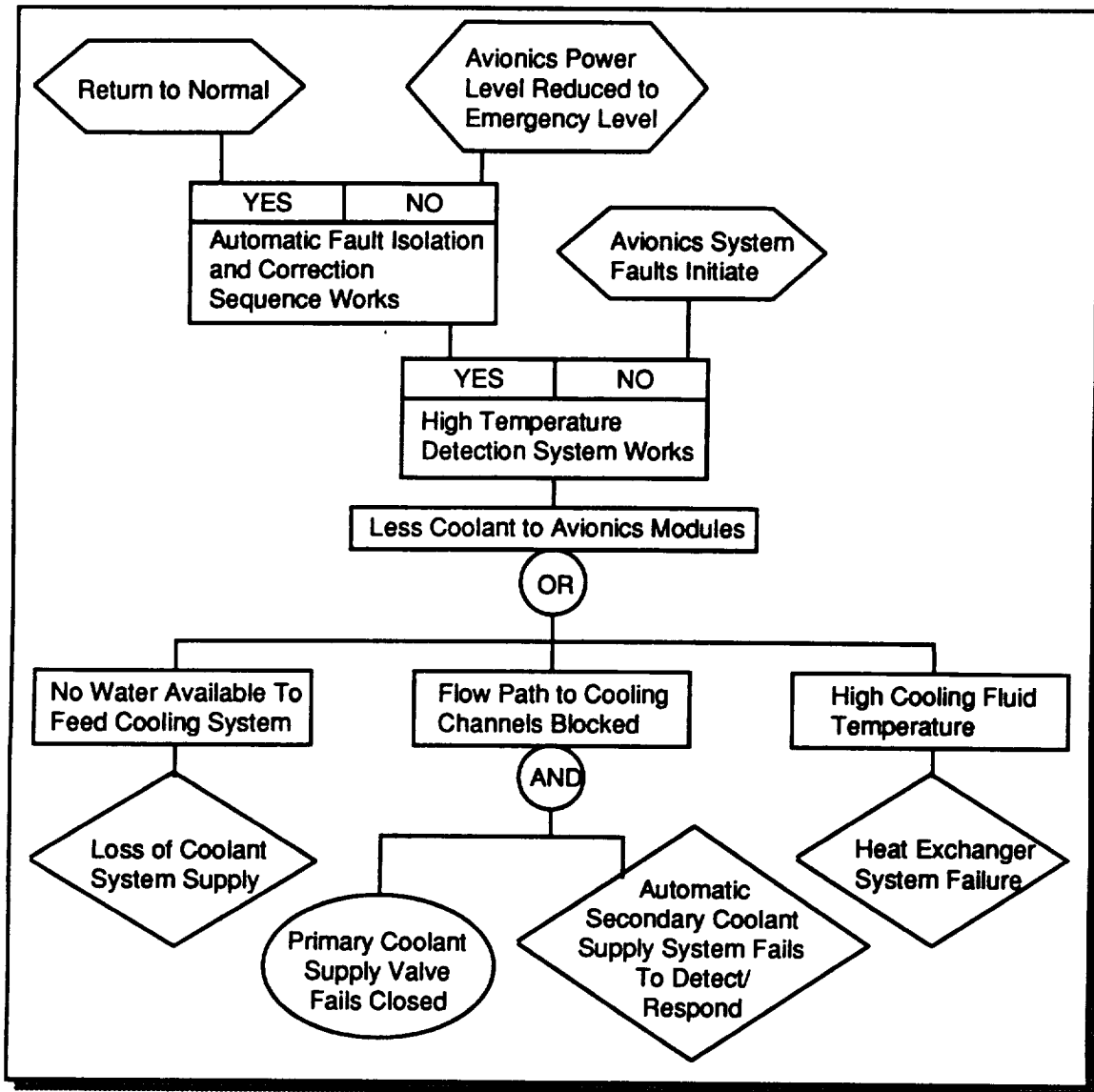


Figure 3.3-8. Avionics Cooling Cause-Consequence Diagram

With the support of system simulations able to determine time constants, the cause consequence diagram can be utilized as the primary tool for a fault propagation analysis. Utilizing a heat transfer transient simulation, the time constants can be determined. Fault detection and containment provisions can be incorporated into the cause consequence diagram and the number of layers of defense against a fault determined. The cause consequence diagram is therefore extremely useful for error and fault containment region design.

Shown in Figure 3.3-9 is the larger context of the fault propagation analysis.

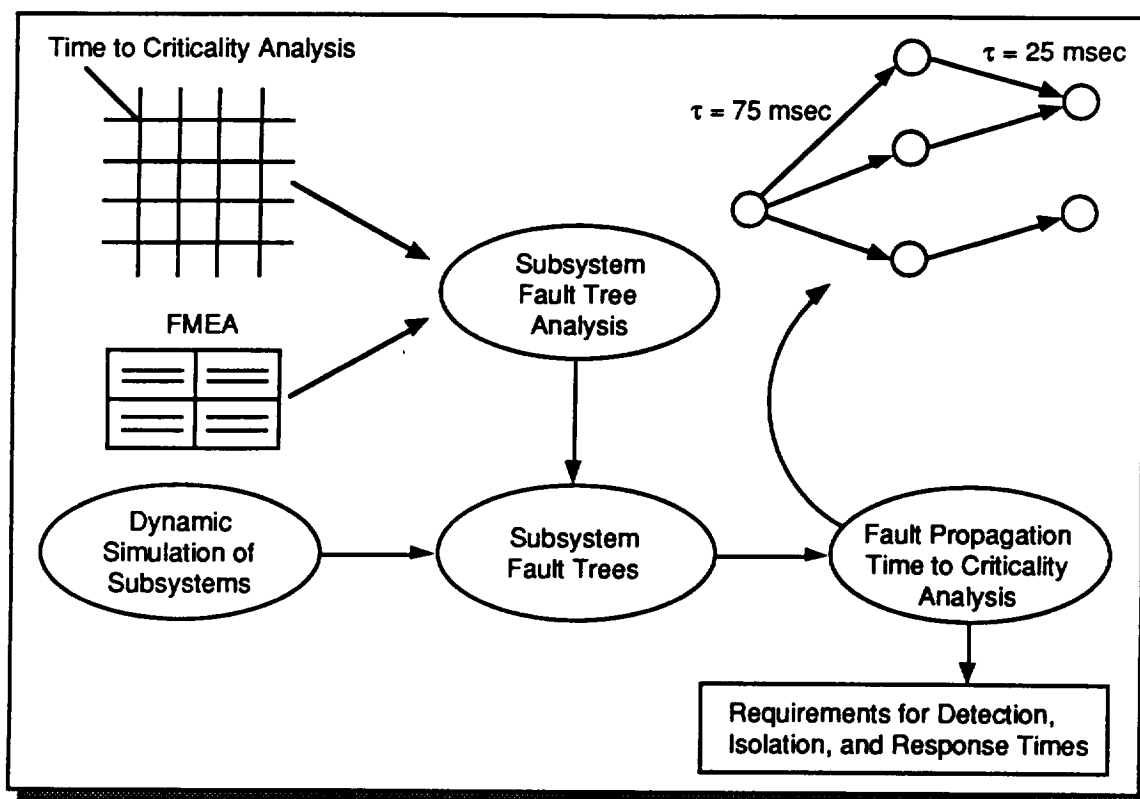


Figure 3.3-9. Time to Criticality for Fault Propagation Process

The analysis is supported by FMEAs and dynamic models of the subsystems. The fault propagation analysis provides the designer with the knowledge of both the pathways of propagation and associated time constants. This provides the knowledge of the fastest potential fault and error propagation path, which in turn drives the design of the fault containment or compensation provisions in the temporal domain.

An example of error containment in a digital electronics device may be buffer register check through parity bit of other type tests. Corrupted data can propagate elsewhere at the data transfer rate from the register. This data transfer rate will define how fast buffer data checks must be executed.

For a rocket engine, fault containment is more difficult. In most cases, rocket engine fault containment apart from engine control system avionics consists of fault detection and engine shutdown. The design information taken from engine fault propagation analyses is usually focused towards determining how fast a fault must be detected and the shutdown response initiated before the turbopumps overspeed or other potentially critical destructive events occur. Shown in **Figure 3.3-10** is a schematic of a typical gas generator cycle rocket engine with a main oxidizer valve closure as an initiating event. Utilizing an engine transient model, the time response trace for the response of key engine parameters to the failure are studied.

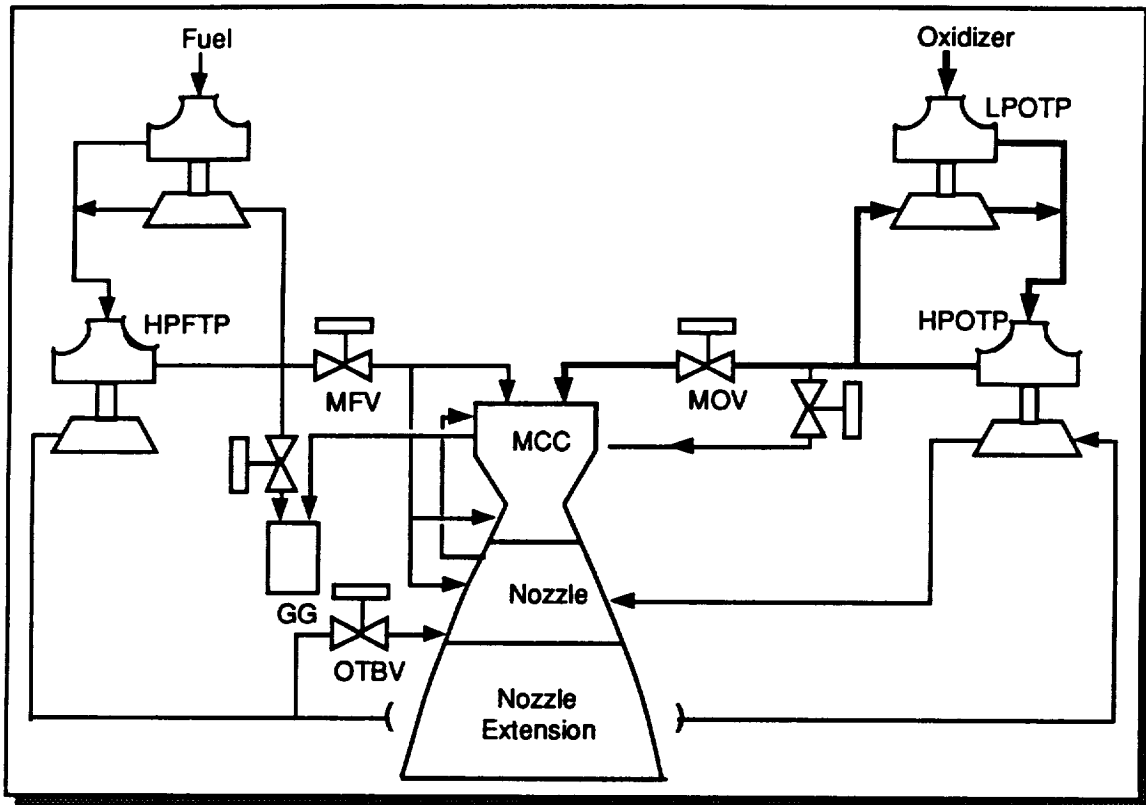


Figure 3.3-10. FTA #2 Main Oxidizer Valve Fails Closed

As shown in **Figure 3.3-11(a)**, engine chamber pressure decays 10% in just over 20 milliseconds, and concurrently, the oxidizer turbopump operating point shifts towards a region of instability as shown in **Figure 3.3-11(b)**. This example of an engine transient response to a fault is typical of a complex level fault propagation analysis, and indicative of the fidelity of simulation needed on some of the more complex subsystems.

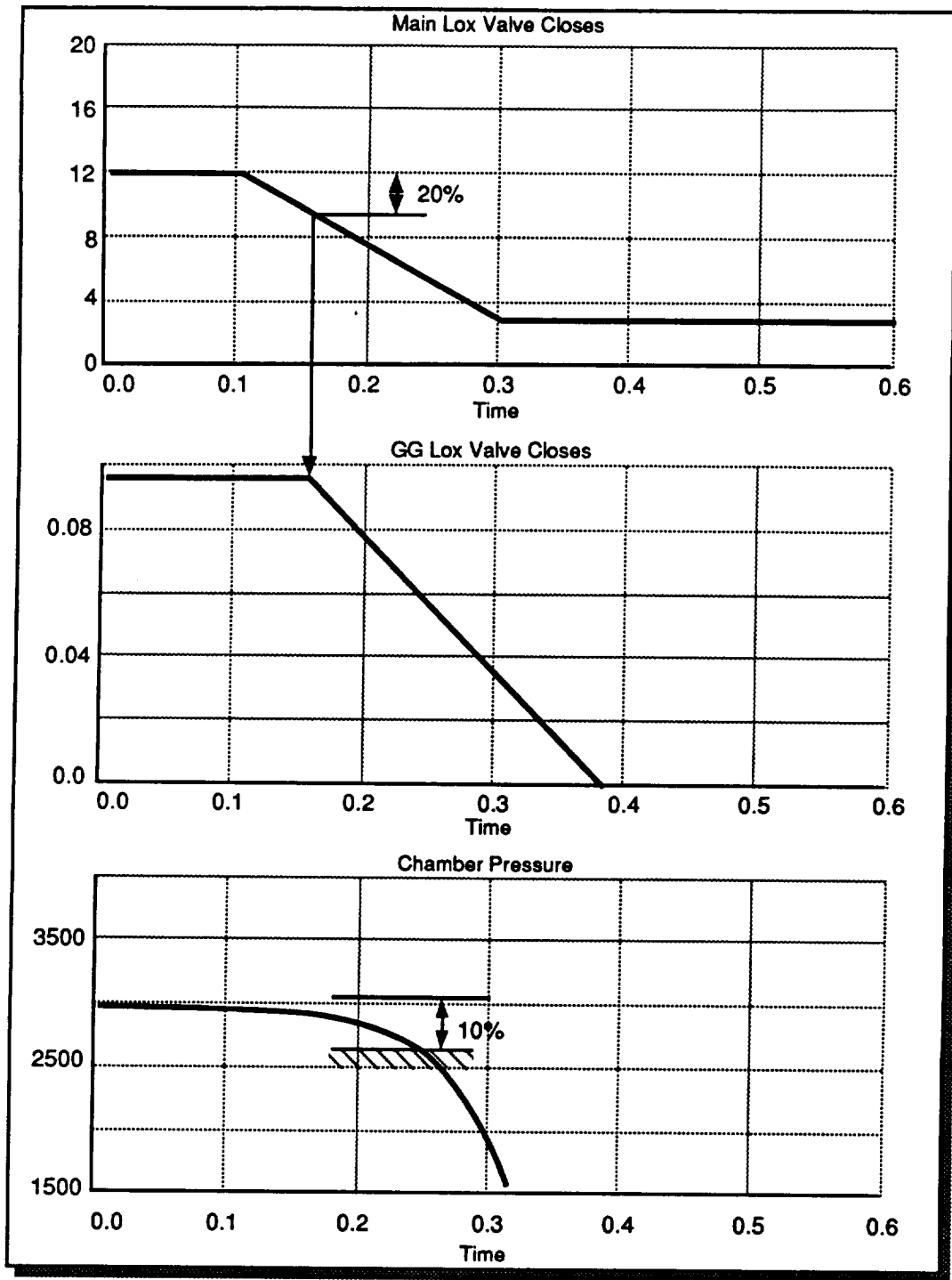


Figure 3.3-11(a). Fault Propagation-Main Oxidizer Valve Fails Closed

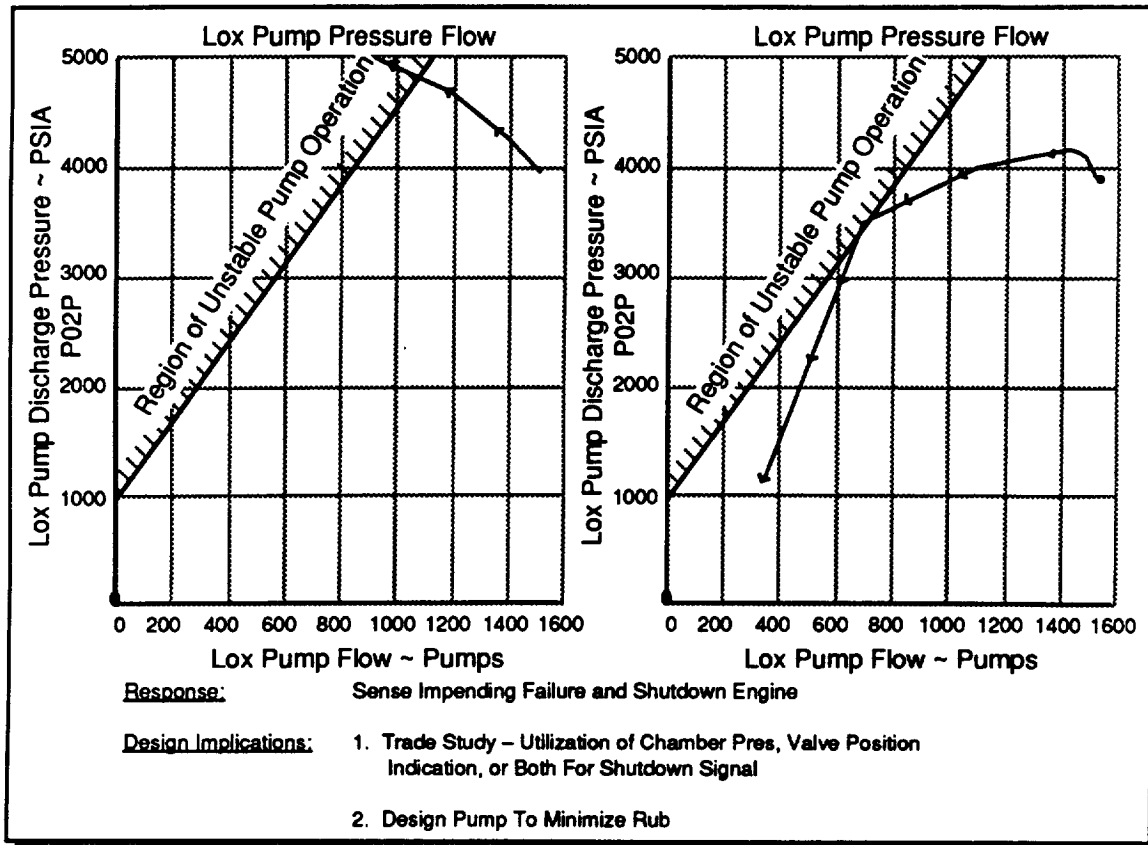


Figure 3.3-11(b). Fault Propagation-Valve Fails Closed

3.3.3.3.3. False Alarm Analyses

Excessive false alarms can destroy the effectiveness of a health management system. Excessive false alarms quickly destroy user confidence, and raise the cry “tear out the sensors and we’ll take our chances”. Many health monitoring systems in aerospace, chemical, and nuclear systems have been plagued with excessive false alarm rates. Shown in **Table 3.3-5** is a matrix comparing actual system health status to perceived system status.

True System Status	Monitoring System Perception		
	Good	Bad	Unsure
Good	Correct Perception of Good	False Alarm	Inconsistent or Incomplete Symptoms for Confident Perception
Bad	Miss (Detection Failure or Out-of-Scope Failure)	Correct Perception of Bad	Inconsistent or Incomplete Symptoms for Confident Perception

Table 3.3-5. System Fault Status Perception

When the health management system perception correctly corresponds to the actual status, detection design intent has been met. Of major concern to the HMS designer are misses, false alarms, and the zone of indecisiveness. The zone of indecisiveness represents cases where symptoms are incomplete or inconsistent to map the symptom set to a fault.

For every detection nominated, both a false alarm analysis and cost effectiveness analysis should be conducted. The basic steps involved in a false alarm analysis are shown in **Figure 3.3-12**. Identification of tolerable false alarm rates should be identified for each detection. The lower the tolerable false alarm rate for a detection, the more stringent design, analysis, and test measures must be imposed. In preliminary design, and again in detailed design, the designer must identify false alarm causes and deal with them.

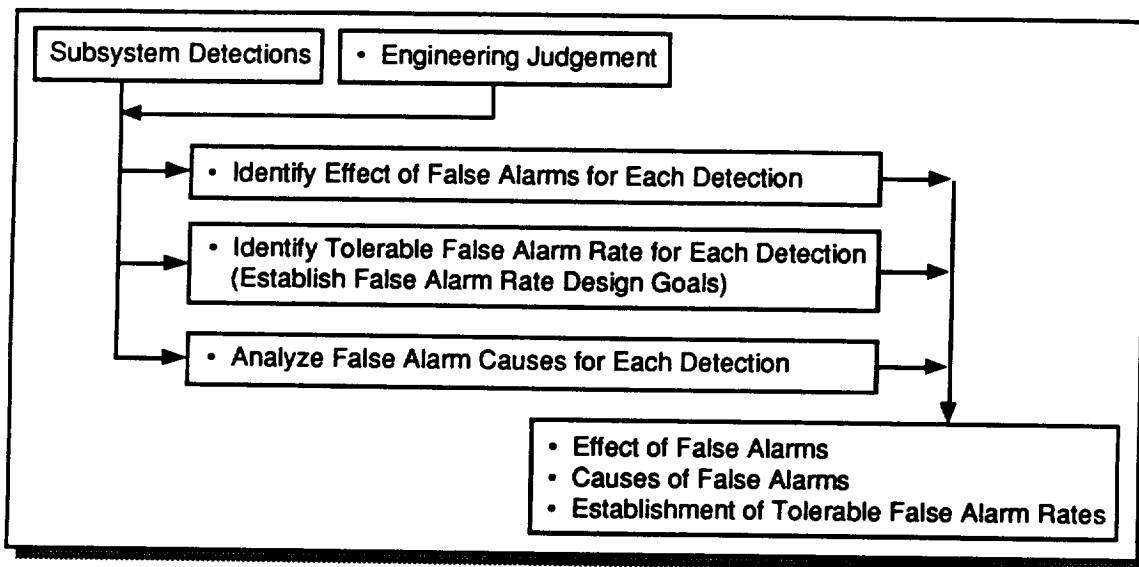


Figure 3.3-12. False Alarm Analysis Execution

Shown in **Figure 3.3-13** are SHM designer options to effect the frequency of occurrence of false alarms. Obviously, these same design variables effect the frequency of fault isolations that fall into the zone of indecisiveness.

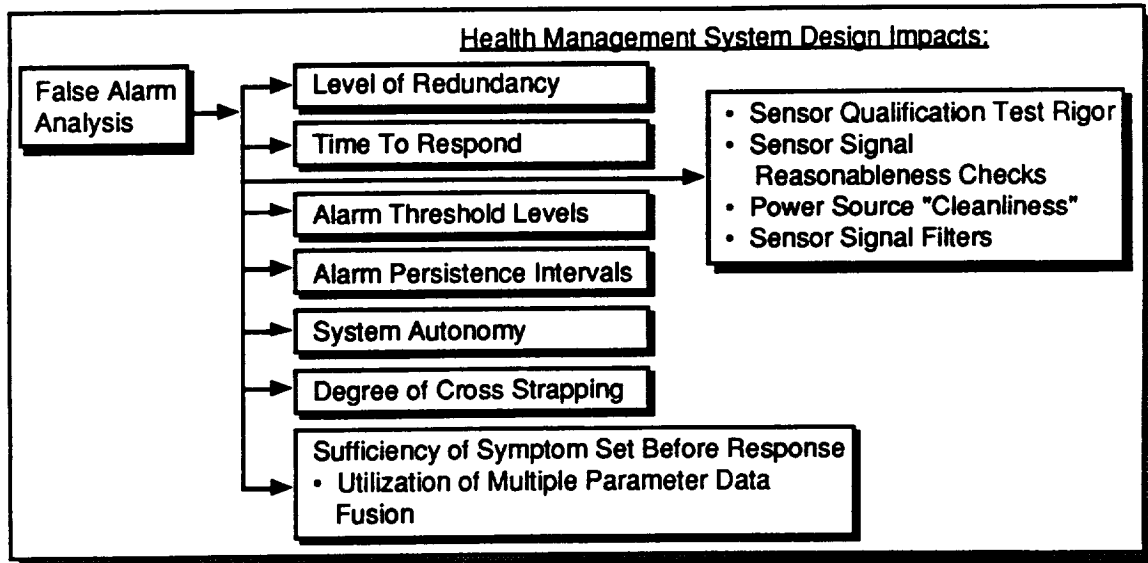


Figure 3.3-13. SHM Design Impact of False Alarm Analysis

Alarm persistence interval, alarm threshold level, and parameter data fusion are three of the most effective design variables to control false alarms. By forcing a symptom set or exceedance to persist for multiple computer cycles before activating an alarm indication, the effect of noise/EMI or other induced signal spikes can be substantially reduced. Raising the alarm threshold level is very effective for reducing false alarms on non-binary signals such as a temperature or pressure. Data fusion, the combining of multiple sensor or parameters into a combined index, can also be extremely effective for false alarm reduction. As an example, if multiple sensors on turbopump outlet pressure and engine combustion chamber pressure all decrease a substantial percentage, the confidence level that an actual significant engine problem exists is much higher than if a single turbopump outlet pressure sensor drops below alarm threshold.

It is possible to compute confidence levels for some detections and isolations. Detection, and symptom to fault correlations can be tagged with numerical indices representing level of confidence. For example, if a fault symptom is derived from a multiple reading of a data fused parameter with no possible ambiguity, a high confidence level of correct detection and isolation occurs. If a fault symptom is shared and maps to several different faults requiring additional measurement analysis to differentiate, a lower level of fault isolation confidence results. This has mathematical similarity to fuzzy logic methods. Confidence level logic greatly adds to system complexity, but certainly is of value for cases where the response is extremely critical. As an extreme example outside the realm of launch vehicles, what is the confidence level that a hostile attack has been initiated by an enemy? If the confidence level of the existence of a fault is low and sufficient time is available for further data correlation to come to a confident conclusion, additional analysis time is highly desirable. Some space systems (not launch vehicles) even have provisions for execution of small tests to confirm and isolate faults.

3.3.3.3.4. Time to Criticality - Preliminary Design Analyses

The objective of time to criticality analyses is to provide designers with time requirements for overall fault detection, isolation, and response (FDIR). The analyses also interweave with the fault propagation studies which help define which faults are compensable. Some faults are inherently non-compensable (such as a primary structural failure), others are non-compensable by virtue of the speed of effects propagation (such as a detonation). Some faults fall in the "gray area" of compensability by virtue of their time to criticality. That is, the time to criticality is so short that the time to isolate detect, isolate, and respond is extremely challenging from a designer perspective. The final decision on whether to implement FDIR for a specific fault is an economic one. Do the required design features to address the fault provide a positive payback relative to the anticipated frequency of occurrence of the fault? (One exception however, could be a man rated system where mandated safety requirement factors override economic factors).

During preliminary design, more definitive design detail enables the top level time to criticality matrix introduced as **Figure 3.2-9** in conceptual design to be completed with substantially less uncertainty. In turn, the time requirements levied upon the subsystems (as discussed in **Section 3.2.3.2.1**) are iteratively updated and improved in fidelity.

Time to criticality analyses now begin a transition from the functional level to the implementation (hardware/software) level. Subsystem functional time response requirements are allocated to specific hardware/software elements. Time to criticality is decomposed as shown in **Figure 3.3-14**.

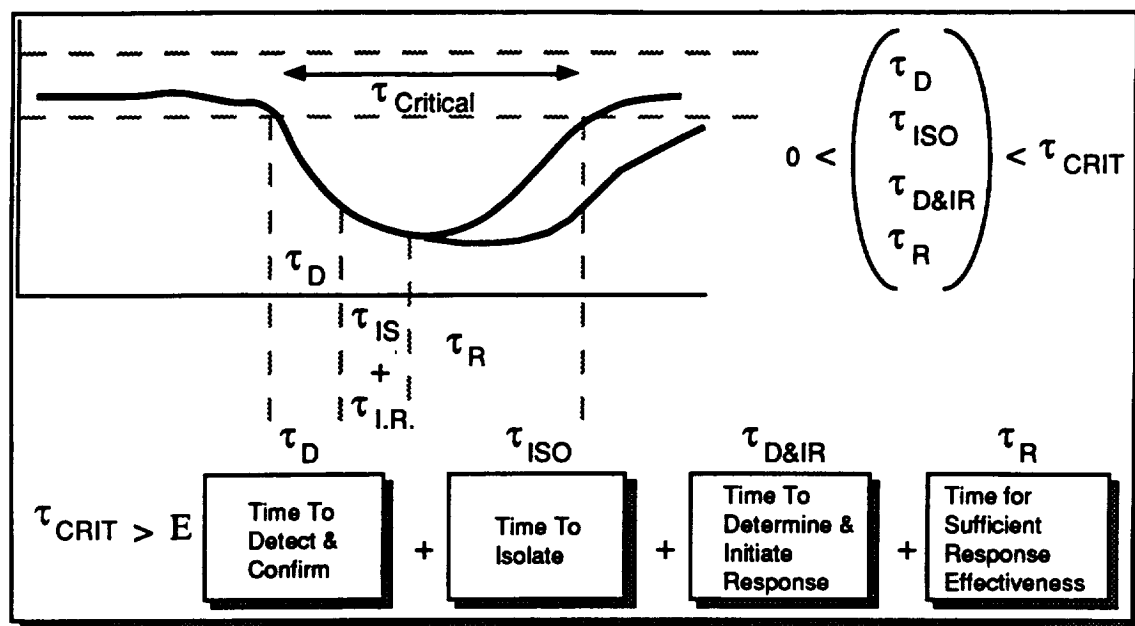


Figure 3.3-14. Allocating Components of Time to Criticality

This figure is a cornerstone of the SHM methodology. The time to detect, isolate, and respond must be less than the time to criticality. Included in detection time is time to confirm the detection. This is important to prevent excessive false alarms. As an example of time to confirm, a redline exceedance can be forced to persist for multiple computer cycles before assuming a true exceedance has occurred. Response time can be decomposed into time to determine and initiate a response, and time for the response (also referred to as the compensation sequence) to take effect to a sufficient degree to bring the system out of the slide towards failure. During a fault scenario, controller constants can be altered to allow more aggressive restorative action. Trade studies are required to best allocate the time to criticality among the four factors shown in the **Figure 3.3-14**. Design issues associated with these trade studies are shown in **Figure 3.3-15**.

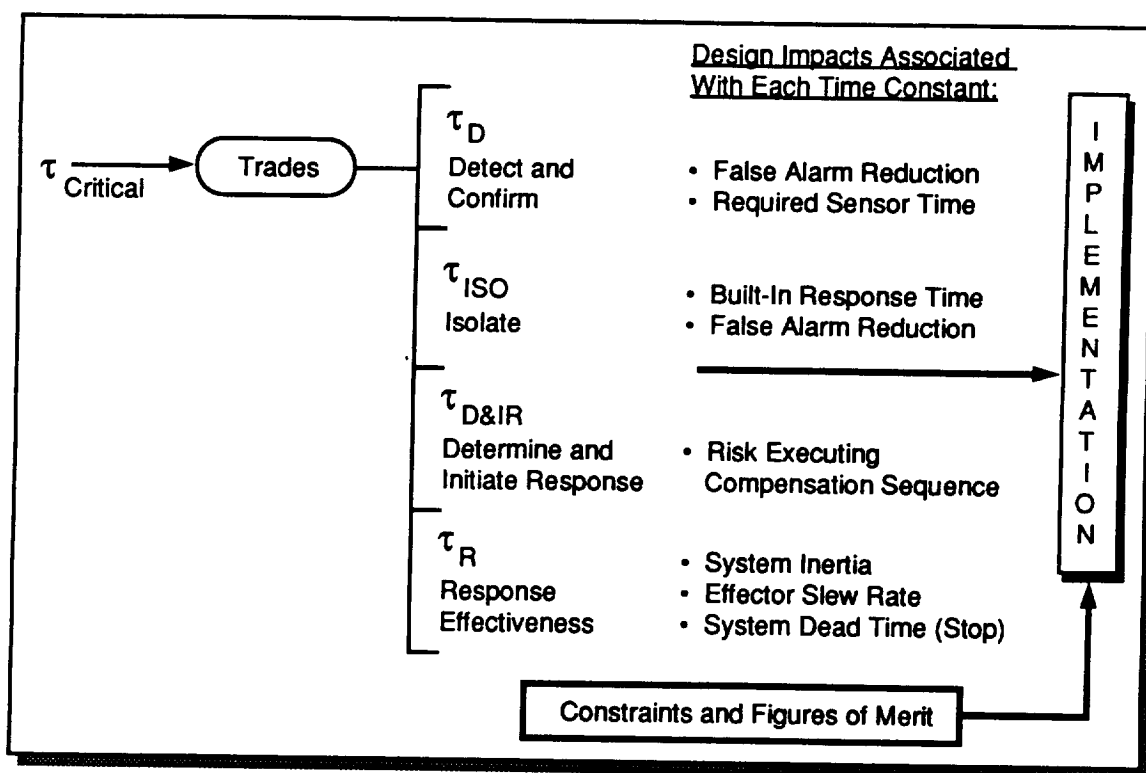


Figure 3.3-15. Designer Breakdown for Time to Criticality

Faster acting effectors and low system inertia can have a dual effect on time to criticality. As a benefit, faster acting effectors can introduce compensatory action more quickly. As a negative, a fast acting effector which slews to an extreme failed position rapidly usually requires faster and more drastic compensatory action. This is an example of how fault tolerance requirements play a significant role in control equipment requirements.

3.3.3.3.5. Cost Effectiveness and Reliability Analysis

One of the greatest challenges to the SHM design is correlating specific design features to reliability and cost. For example, what is the cost and reliability of a triplex sensor arrangement without cross strapping versus a triplex arrangement with cross strapping? Is the reliability improvement expected for one concept worth the extra cost? What is the uncertainty of the analysis? Are software complexities being adequately costed? The sophistication, correctness, and user friendliness of tools for cost and reliability estimation of different design options is crucial during preliminary and detail design.

Essential to cost analysis is an agreed upon set of groundrules. Inflation factors, labor costs, cost of money, and a myriad of other cost factors must be agreed to before preliminary design begins. A common set of SHM cost modeling tools is much needed.

Schedule constraints will override many decisions based on cost alone. Timetables for development and risk factors associated with unproven concepts will eliminate many options.

Cost Effectiveness Analysis

Aside from man-rating requirements or other system design mandates, this SHM design methodology requires that health management system design features “buy” their way into the design through demonstrated cost effectiveness. This approach prevents parameter lists and associated SHM costs from growing to unwieldy size (better known as the metrologist syndrome). Cost effectiveness analysis also acts as a referee between subsystems to prioritize what should be eliminated or added to stay within cost constraints. To the greatest extent possible, the quantitative justification of the SHM design approach should be utilized. Although noble in intent, the philosophy is complicated by disagreement on assumptions used for cost analysis, and the complexity of estimating the cost and reliability of avionic architectures, particularly those with substantial amounts of fault tolerant software. A prioritized SHM accommodation measures list is only of value if the spread between elements of the list is greater than the uncertainty of the numbers that went into the analysis. There are wide variations of the uncertainty levels from subsystem to subsystem.

The first step in SHM cost effectiveness analysis is for the customer and design team (typically several companies) to agree on the elements to be factored into cost analysis. Shown in Table 3.3-6 are some of the top level factors that should be considered in SHM cost modeling. If life cycle cost is used, the groundrules for the elements of recurring and non-recurring costs must be identified. Since operations and software costs play a major role in overall SHM cost analysis, it is imperative to fairly cost hardware maintenance, spares, logistics, training, software upgrade, ground support equipment, computer upgrades, and many other factors, which if ignored, can substantially skew the data.

Benefits:

- Reduced Flight Losses
- Reduced Mission Loss Downtime
- Expedited and Improved Efficiency Assembly, Integration, and Maintenance
- Improved Confidence and Situational Awareness for Pad Operations
- Life Usage and Maintenance on Condition for Reuseable Components

Penalties:

- Development Cost
- Production Cost
- Operation and Support Cost for Hardware and Software
- False Alarms (In Any Phase)
- Weight Penalty

Table 3.3-6 Major Factors for SHM Cost Benefit Analysis

If specific health management system features are assumed to contribute to expediting ground checkout, an operations model is needed to translate these features into dollar savings. Particularly where a multitude of factors contribute to ground checkout cost savings, apportionment of the cost savings to specific system elements is a difficult task. If systems level bookkeeping is not done, several groups can easily take duplicate credit for the same savings.

The mission model over which the non-recurring costs are distributed dramatically effects the cost analysis, as do number of launch sites, infrastructure, labor cost assumptions, cost of money, etc. How is the cost of unreliability handled? What are the assumptions used for downtime avoidance and the cost of downtime? Instrumentation development costs can be substantial. For sensors used for both control and health management, to what category are the costs attributed? Subtle bookkeeping rule changes on what is charged to health management and what is charged to other categories can greatly bias cost trade studies.

Once the top level cost groundrules are established, detail design rules are required to translate specific design implementations into costs. For example, what complexity factor should be added to a cross strapped triplex configuration compared to a non-cross strapped triplex configuration? Is the cross strapping implemented mostly on a hardware or software level? Is sufficient funding set aside for testing and verification, particularly to study interaction effects? If two companies estimate the cost of different hardware implementation approaches for a trade study, have they utilized uniform costing assumptions?

In summary, the cost analyses of the benefits and penalties of health management system options provide the basis for determining what features can “buy” their way into the design. If items cannot be justified by economic benefit analysis, other requirements such as safety mandates must be identified as justification. (Man-rating requirements often justify design features which don’t qualify from a pure economic argument approach). As a minimum, a relative ranking of health management system design feature payoff can be assembled.

Reliability Analysis

Reliability analyses are closely associated with the SHM cost effectiveness analyses. The reliability analysis results are usually provided as inputs to the cost effectiveness analyses, with the realization that a given increment of reliability change translates into a specific cost effect.

There are many levels of reliability analysis. At the system level, an estimate of reliability of subsystems is needed. To achieve design reliability goals, different design approaches are synthesized and subjected to trade studies. To achieve the required reliability, the designer is faced with the choice of an approach relying on extremely high inherent reliability of single string elements or lower individual element reliability and a redundancy approach featuring switchout/shutdown/voting. The overall launch system invariably contains a combination of non-redundant primary elements and different levels of redundancy within subsystems and line replaceable units.

Historical data are necessary to estimate the reliability of system design concepts. Rome Air Development Center has supported the development of failure rate databases for electronic components. Similar databases exist within industry and government for non-electrical components. With rapidly changing technology, the key to effective utilization of databases is to understand the nearness of applicability of the data to the design, an often difficult challenge. Historical data is valuable for bracketing estimates.

Key to good reliability analyses are:

- An understanding of the applicable environment and appropriate correction of data to account for the environment. Knowing the validity of data estimates by an appeal to commonality and complexity factors.
- An understanding of the uncertainty of the reliability estimates.

- Correctly identifying tall poles and dominating factors
- Understanding the role of software reliability and human factors in the overall reliability estimate
- Searching for hidden common causes of failure and overlooked interconnections effecting reliability
- Utilizing verification tests to confirm critical reliability assumptions

Tools for Reliability Analysis

General classifications of tools to analyze system reliability include fault trees, failure modes and effects analyses, success diagrams, cause-consequence or cause tree methods, truth table methods, gathered fault combination methods, directed graph models, and state-space / state change models often incorporating Markovian techniques.

Specific tools for reliability analysis are described in **Appendix D**.

Software Reliability Analysis

Designing and assuring software reliability is crucial to the HMS design. Since the HMS software is often embedded and integrated into the general system software, HMS software reliability plan must be developed in conjunction with the overall system software reliability plan.

Experience has shown that software can have faults and still function acceptably nearly all of the time. This occurs because the same logic path can manifest an error for some but not all input data.

It is important to acquire first indications of HMS software reliability as early as possible. Hopefully, laboratory software from small HMS demonstrations and similar projects can enable this work to begin in the HMS preliminary design phase. As the first HMS test code is developed in preliminary or detail design, software reliability metrics specific to the particular implementation can be acquired. The first indications of implementation specific software reliability are acquired from complexity measurements and augmented with test-related measurements.

The next step in software reliability prediction is model development. Model types are typically [VILLEMEUR 91]:

- a) "perfect debugging" model
- b) "imperfect debugging" model
- c) "Random debugging" model
- d) "Bugs with different occurrence rates" model
- e) Parametric models

Numerous texts and resources are available to guide the programmer in methodology for software reliability. Two of these are Chapter 6.1 of [AND/DORF 91] and Chapter 17 [LLOYD-LIPOW 84]. This field is evolving quickly.

3.3.3.4. Preliminary SHM Design

3.3.3.4.1. Development of the Fault Accommodation List

As noted earlier, the preliminary fault accommodation list begun in the conceptual design phase (and introduced as **Table 3.2-5** in **Section 3.2.3.2.2**) is carried to a significantly higher degree of fidelity in the preliminary design phase. The FMEA is a major input to the development of a more detailed fault accommodation list.(shown in **Figure 3.3-16**). This table is central to the SHM methodology.

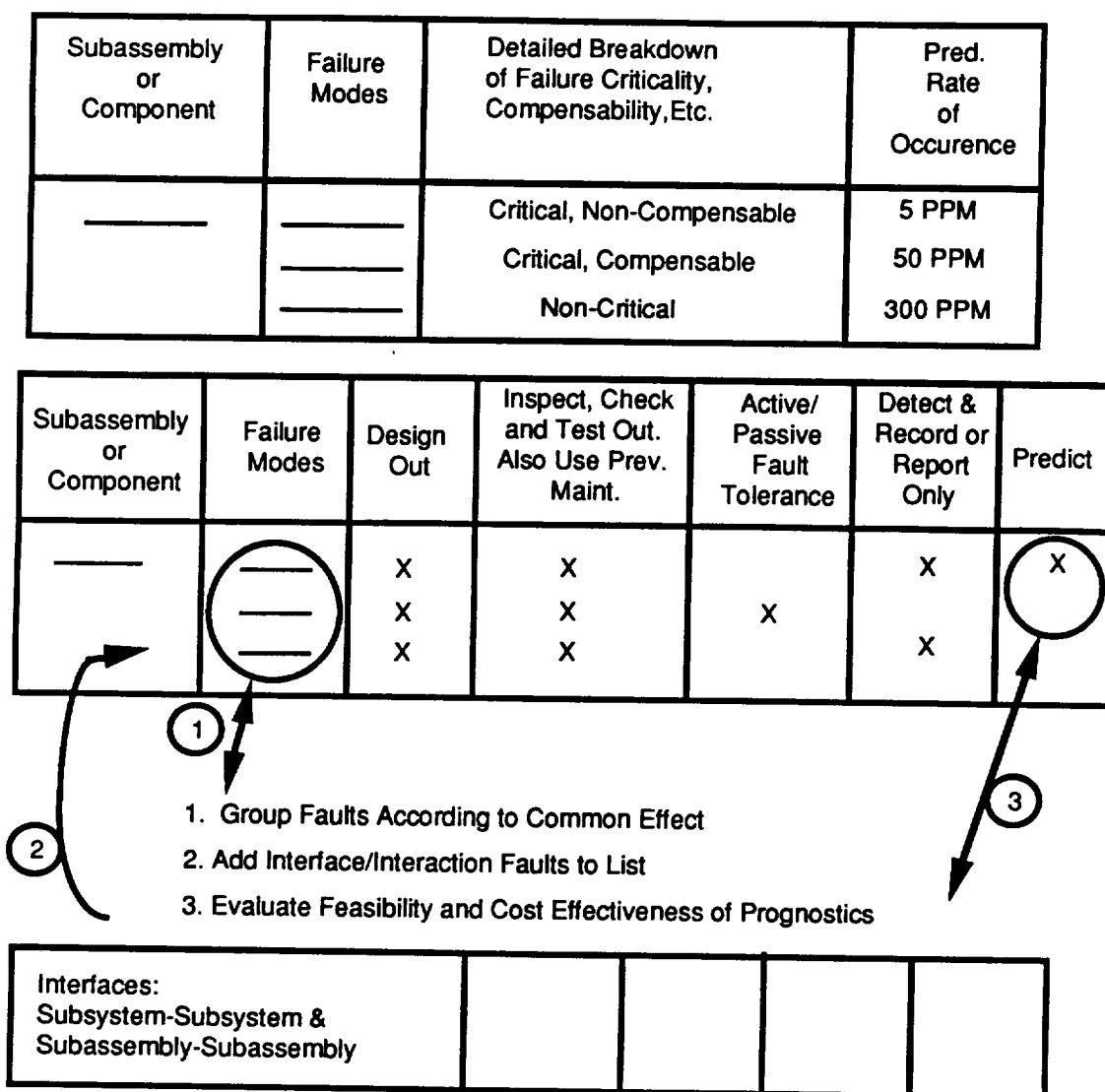


Figure 3.3-16. The Fault Accommodation List

Additions/alterations to the fault accommodation list at this time are:

- 1) Grouping of faults according to common effects.
- 2) Focused attention to the inclusion of subsystem/interface interaction faults.
- 3) Identification of faults for which prognostics/fault prediction is practical and likely to be cost effective

It is important to keep certain classes of faults in mind while developing the fault accommodation list, especially with regards to identifying the human made, external, and interaction faults. The fault taxonomy tree suggested by A. Avizienis, and shown in **Figure 3.3-17** is an excellent tool to utilize in this instance. Human made faults are those created in design, specification, and human interaction such as process and maintenance. Since most of these kinds of processes are only vaguely conceptualized at this time, these classes of faults will appear in **Figure 3.3-16** in very generalized form at this point in design.

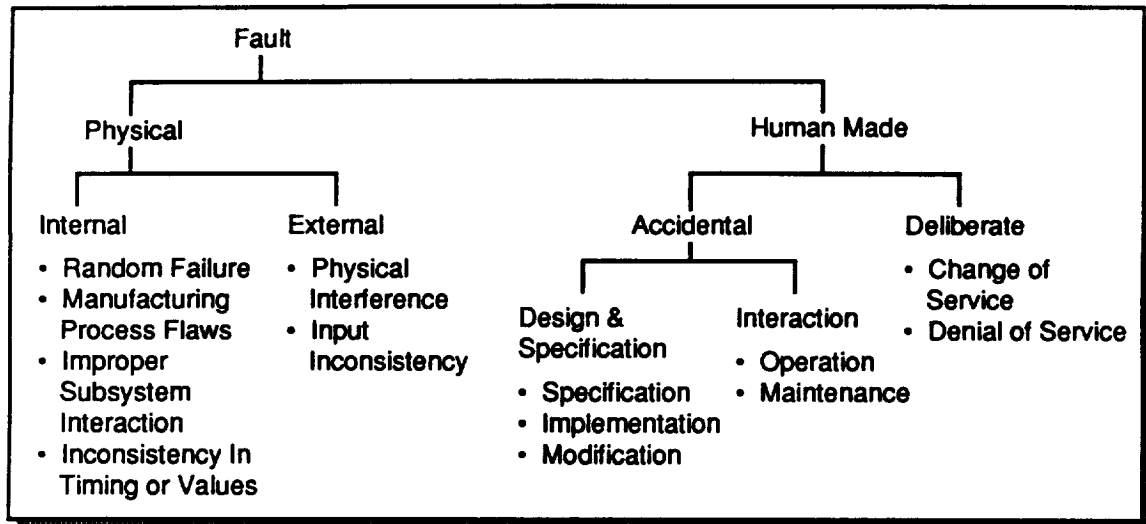


Figure 3.3-17. Fault Taxonomy Tree

FMEA/Fault Accommodation List - Failure Criticality

The criticality column of the fault accommodation list is now expanded. The failure modes and estimated rates of occurrence must now be numerically decomposed into categories as shown in the upper part of **Figure 3.3-16**

3.3.3.4.2. ECR/FCR Refinement

Better subsystem detail during preliminary design enables more detail to emerge in the formulation of layered (hierarchical) fault and error protection containment regions. (Refer to **Section 3.2.3.2.4** for the ECR/FCR aspects of conceptual design.) As noted in ECR/FCR conceptual design, ECR/FCR design depends upon an adequate strawman fault set, so the design effectiveness can be assessed against different types of faults. Error and fault containment design must consider both the physical region domain and the time dimension. Likewise, error and fault containment must consider the digital, analog, and mechanical domains, each of which is characterized by distinct ECR/FCR design techniques.

Error and fault containment with respect to digital electronics is typically embedded within a larger sequence of fault handling actions as listed in **Table 3.3-7 [SIEW 91]**.

1. Fault Confinement
2. Fault Detection (Note confinement frequently precedes detection)
3. Fault Masking
4. Retry
5. Diagnosis/Isolation
6. Reconfiguration
7. Recovery
8. Restart
9. Repair
10. Reintegration

Table 3.3-7. Typical Digital Fault Handling Sequence

For digital system ECR/FCR, the full sequence listed must be considered. Successful containment and recovery depends upon all the actions listed.

Table 3.3-8 is a greatly simplified and hypothetical example of a 5 layered error containment region for avionics application taken from [SIEW 91].

Level	Typical Error Sources	Typical Error Recovery Technique	Typical Error Response Time
Application	Incorrect Coding of Algorithm	Reasonability Checks	10E-01
Operating System	Incorrect Design	Consistency Checks On Data Structures	10E-03
Macrocode	Alpha Particles Flip Memory State	Memory Protection Violation	10E-04
Microcode	Race Condition	Error Coding	10E-06
Software	Environmentally Produced Transient	Replication	10E-07

Table 3.3-8. Hypothetical Five Layered ECR (from[SIEW 91])

The higher the error moves up the layered defense, the more elements of the system are affected. In turn, as the error moves up the hierarchy, the greater the complexity and time required for isolation and mitigation. The HMS ECR/FCR design must be analyzed and based on a hierarchical perspective. Of great challenge to the designer is determination of how complex a multi-layered ECR/FCR defense should be established. Cost effectiveness analyses are appropriate, but dependable data on coverage and time latency for different ECR/FCR options is usually difficult to acquire.

Table 3.3-9 lists some of the typical techniques used with digital processors for fault confinement and detection.

- Fault Confinement:
 - Liberal Use of Fault Detection Circuits
 - Consistency Checks Before Performing A Function
 - Multiple Request/Confirmation Before Performing A Function
- Fault Masking
 - Voting With Three or More
- Fault Detection Techniques
 - Duplication
 - Error Detection Codes
 - Consistency Checking
 - Self Checking and Fail Safe Logic
 - Watch Dog Timers and Timeouts

Table 3.3-9. Typical Techniques For Digital Processor Fault Confinement and Detection

An ambiguity zone is considered the domain/region to which the fault/error is known to exist in by virtue of the currently available information. If the fault/error can be bounded to a domain/region for which the response is unique, then the degree of isolation is sufficient from an operations perspective. (Although from a designer perspective, the ambiguity zone is never small enough from a post incident analysis perspective). The usefulness of the gathered fault combination method (GFCM) to collect faults according to common effect is again apparent. The usefulness of design practices to force the effects of failure into a small set of categories is also apparent during the FCR/ECR architecture formulation.

For a single processor, self checking is usually employed as a detection technique. For a dual set of processors, comparison is utilized for "mutual suspicion" cross checking. For a triplex or higher configuration, voting is usually employed. The higher degrees of redundancy provide a higher degree of coverage but also cost more to employ. Comparative cost effectiveness analyses are useful, but difficult to conduct with the degree of certainty desired to eliminate designer qualitative judgment.

Error and Fault Containment Regions for Mechanical/Propulsion Systems

For liquid rocket engines, the entire engine is typically the fault containment region. Apart from the engine controller, engine shutdown is typically the only response to engine component failure, and the shutdown response of value only if there is the engine out option, or, a catastrophic failure consequence avoided for a manned vehicle.

Thicker casings on turbopumps can reduce the probability of a turbopump failure destroying adjacent systems. Similarly, fault containment can be improved by judicious physical location selection for mechanical equipment that prevents cross system genocide. For valves, double seals and double valving arrangements provide multi-layered fault containment.

Analog FCR/ECR

In the analog domain, capacitors, circuit inductance, circuit breakers, and fiber optic lines are techniques employed to control the spread of faults. Fiber optics are particularly effective for EMI, stray current, and static discharge effect reduction.

ECR/FCR Summary

In summary, during preliminary design, the ECR/FCR factors that must be considered are listed in **Table 3.3-10**.

1. The Complete Fault Type Set for ECR/FCR Design (Intermittent, Interaction, Near-Coincident, Timing, Etc.)
2. Physical and Time Domain Aspects of ECR/FCR
3. Design Practice for Mechanical and Analog Fault Containment
4. The Full Sequence of Actions for Fault Detection, Isolation, and Response. ECR/FCR Must Be Done From the Full FDIR Perspective
5. Hierarchical Layering of ECR/FCR
6. Analysis of Ambiguity Zone Breadth and the Cost Effectiveness of Smaller Ambiguity Zones.

Table 3.3-10. Factors To Consider In ECR/FCR Design

3.3.3.4.3. Fault Prediction, Detection, Isolation, Response Implementation

All of the fault accommodation columns except design out require a detection, isolation, and response plan. For ground tests/checkouts, there are built in tests, tests conducted with support hardware (such as the Harris CORE system or Martin Marietta PAGE system), non-destructive evaluation tests (such as computer tomography scans of solid rocket motors), visual inspections, interface tests, and others. Regardless of the type of test, a response plan needs to be formulated. For ground tests, there are usually combinations of computer directed responses and points of human interaction in the decision process of how to respond. This response logic also applies to flight failures which are of such benign character that they are simply detected and reported, as reflected by the column titled "detect and record or report only" of Figure 3.3-16.

Merger of Top Down and Bottoms Up Viewpoint of Faults

The most challenging fault response determination category is that of flight mode active fault tolerance. The time to criticality for faults in flight mode is often short, and critical category faults are more numerous. At this time, there is a need to merge the top down functional fault matrices (an example of which was introduced as Figure 3.2-11), and the bottoms up FMEA fault matrices. The top down approach is important so that subsystem interaction faults and system level faults are not overlooked when individual subsystem FMEAs are assembled. As discussed in Section 3.3.3.2, the gathered fault combination method (GFCM) of analysis is an excellent technique to group the faults together in the fault accommodation list which produce the same effect. GFCM fault "gathering" is useful for the design of error and fault containment region partitioning, verification, and the design of accommodation provisions.

Interface and Interaction Faults

Another extremely important action to execute at this time is addition of interface and subsystem interaction faults to the fault accommodation list shown in Figure 3.3-16. In moderately complex system architectures, there is generally too much emphasis on testing of elements as stand alone entities when in practice, a high percentage of faults appear as interaction problems across ECR/FCR interface boundaries.

Refinement of Prognostics In Fault Accommodation List

Prognostics were first discussed in Section 3.2.3.2.5. At this time, fault conditions which can be predicted before they occur should be identified and entered into the chart of Figure 3.3-16. The fault prognostic candidates should be considered nominees for technical feasibility study and trade study evaluation for cost effectiveness. Prognostics are more applicable to long life equipment such as ground equipment or recoverable resources, and rarely have a significant role in disposable launch vehicles.

Detection

As illustrated in Figure 3.3-14 previously introduced, detection is the first part of the chain of detection, isolation, and response. The time to criticality estimates for each fault allows the designer some liberty as to how to use the time available most effectively. More time spent confirming a detection and assuring the isolation has been made to the proper fault grouping is valuable if the time to criticality provides the designer this freedom.

Detection confidence can be improved by using data fusion, looking at indices which combine several detections indicative of the existence of a particular fault.

As an example, both chamber pressure and oxidizer injector pressure should be effected if the oxidizer turbopump is malfunctioning. Detection confidence is also improved by forcing a symptom set to persist for two or more computer cycles, reducing vulnerability to single cycle upsets.

Isolation: Symptom/Detection to Fault Mapping

Figure 3.3-18 is a sample of a generic symptom to fault mapping, or fault isolation.

		Symptoms/Detections					
			S1	S2	S3	S4	S5
Group 1	F1	F a u l t s	x	x			
			F2		x		
F3					x		
F4				x			
F5				x	x		
F6						x	

Figure 3.3-18. Detections Are Mapped to Fault

Ideally, parameters can be chosen so there are unambiguous symptom to fault correlations. In the example, detection/symptom S1 uniquely maps to fault 1, and symptom/detection S5 uniquely maps to fault F6. Symptoms S2, S3, and S4 do not uniquely map to a single fault however. Isolation to a single fault is usually not necessary from a fault response perspective. As long as isolation has narrowed the fault to a grouping for which the same response is called for, isolation is sufficient from a response perspective. However, it is desirable from a post incident perspective to know what specific fault caused the problem, not just what fault grouping produced the problem. If the loss of a single sensor clouds the ability to isolate to the correct fault grouping, the design is weak. The value of the gathered fault combination method is again apparent.

Figure 3.3-19 provides a guideline for the design of a detection, isolation, and response plan for flight mode faults.

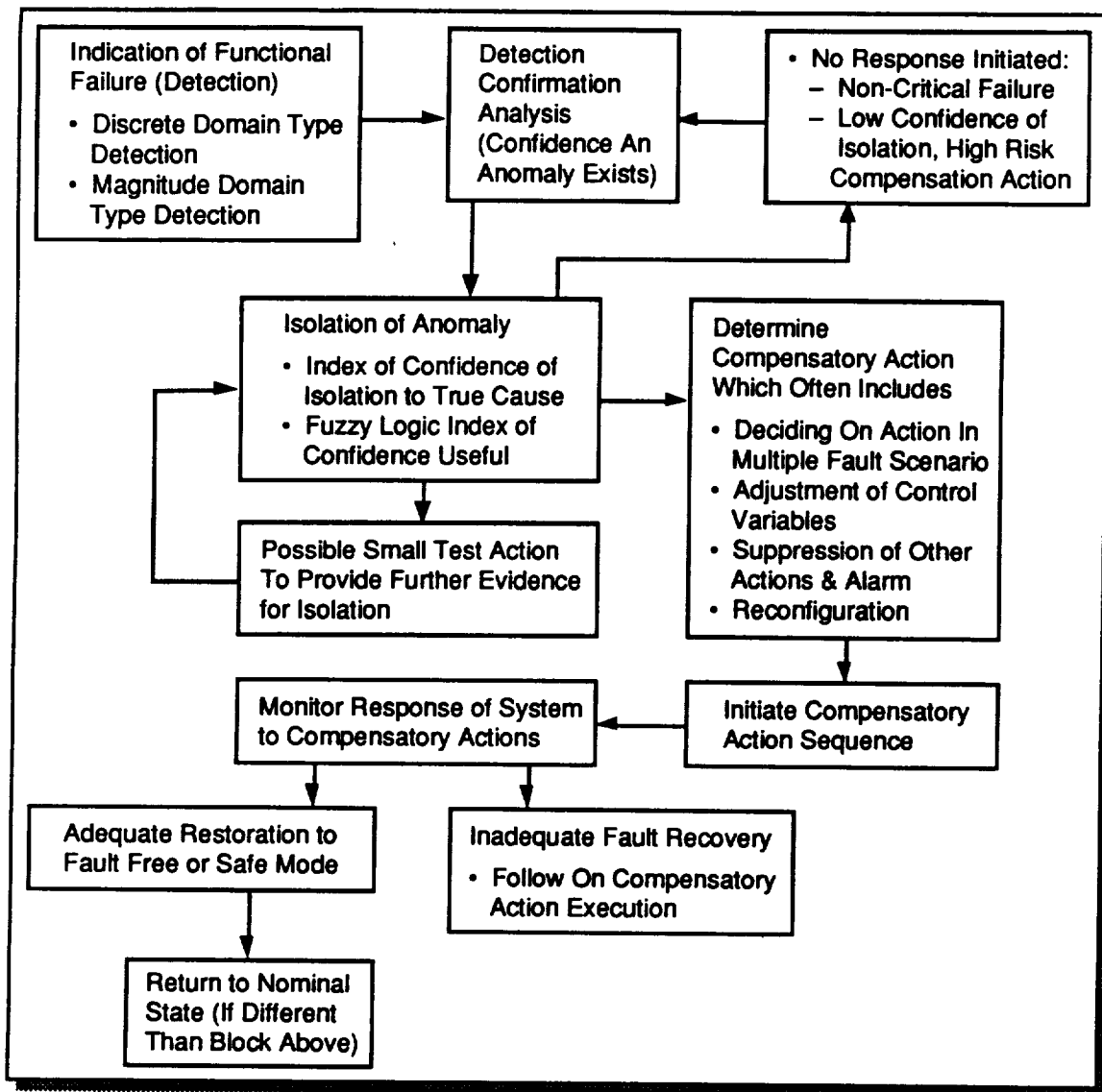


Figure 3.3-19. Fault Response Sequence

Atypical for launch vehicles, but possible for many space vehicles, and reflected by a block in Figure 3.3-19, is the ability to execute small tests within the vehicle during flight to gain additional information for correct isolation.

Response

If the isolation algorithms have reduced the fault ambiguity to a specific fault grouping for which the same response is called for, the health management system design is well on the way to success. There is always a risk involved with the execution of a fault response. The compensatory sequence may not work, or may involve a series of complex operations vulnerable to failure, particularly when executed in an environment where at least one fault or failure has occurred.

Once the system state vector has slid into a fault condition, strange effects and conditions can arise that were overlooked or unanticipated by the designer. Figure 3.3-20 illustrates a typical feedback control loop set. Disturbances to the loop can often be externally induced, or induced by failures within the subsystem or elements of the control loop itself. This requires that designers explore the fault domain with even more care than the normal domain of disturbances.

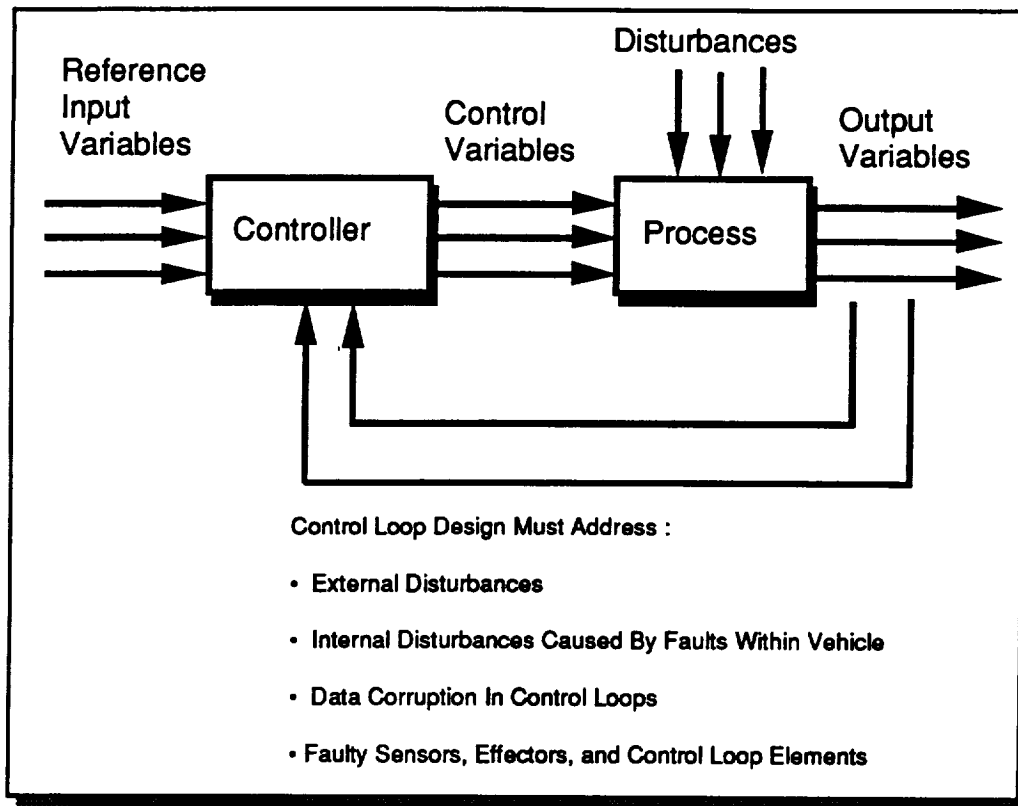


Figure 3.3-20. A Typical Feedback Control Loop Set

As shown in the “determine compensatory action” block of **Figure 3.3-19**, suppression of other alarms and temporary readjustment of control loop gains and constants is often required until adequate restoration to a fault free or safe mode occurs. In the process of restoring the state vector from a fault condition state to normal operation, the path of state vector restoration is extremely important to avoid setting off additional alarm conditions and driving the system into a state of higher entropy even further from normal operation.

If error and fault containment regions have been well designed, fault propagation is minimized. There are certain subsystems relatively immune to finely partitioned fault containment however, such as engines. The entire engine tends to be a fault containment region with use of engine out. Where fault containment is more global, responses are likewise more global.

Response for Multiple Independent Faults and Complex Fault Cascading

Multiple independent faults are generally a very rare event, and not considered in most health management system designs. Where faults cascade, the time domain must be carefully considered in design, particularly where a lower tier triggered fault calls for a response in conflict with the initiating event. This situation is rare on a launch vehicle, but more probable on a space vehicle. If this possibility is uncovered in the analysis, the designer must design the response logic accordingly.

3.3.3.4.4. SHM Implementation Issues

Ground System Health Management

The ground system health management design is interwoven with the entire launch vehicle systems design process. In this section, some of the particular factors to check in the ground system design are discussed.

Requirements & Design Goals

The requirements for ground system health management systems are usually substantially different from the flight vehicle. In general, ground system faults are less critical than flight mode faults. Availability tends to be the more dominant attribute of dependability when dealing with ground systems.

Weight is usually of minimal significance in ground design, and accessibility and maintainability paramount. Since ground systems often experience hundreds of cycles spanning more than a decade, long term stability, drift, and trending for maintenance on condition are important. Commonality and suitability for modification are important design considerations. Design for minimizing recurring operations and maintenance costs should dominate ground system health management requirements.

Time to Criticality - Ground Systems

The concept of time to criticality still applies to ground operations, but the consequences are typically operational delays. Ground time delays are often step function in nature, with minimal impact if addressed quickly, but sharply increased negative consequences if the delay becomes the critical path and impacts other concurrent or serially scheduled events.

3.3.3.5. Simulation/Test Bed Design

As discussed in Section 3.2.3.3.1, our methodology emphasizes early identification of simulation and test bed requirements including requirements for later refinement and expansion as more becomes known about the System. Thus at the preliminary stage of design the simulation and testbed design actually begins in earnest based on these requirements. However, the actual test bed and simulation needs changing as the system design progresses. Simulation and test bed design requirements must be updated using the preliminary FMEA, preliminary subsystem design development and based on preliminary use/testing of the designed simulations. At the preliminary design phase, the level of simulation and test bed design completed in some cases will mirror subsystem design levels. In other cases simulation/testbed design must precede subsystem design because these developing simulations and test beds will be used as design analysis tools,

3.3.3.6. Detail Design Requirements

At this stage in the design process HMS requirement development seeks to:

- 1) reduce the ambiguity of existing preliminary design requirements such as those shown in Table 3.2.6
- 2) and/or identify missing requirements.

Knowledge of the system has increased greatly. Requirements must be documented and kept current, accurate and accessible to everyone involved in the engineering design process.

3.3.4. Preliminary Design Phase Summary

Table 3.3-11 summarizes the major design activities of the HMS preliminary design phase.

- Generate Overall Fault Matrix (Prelim. FMEA) for Subsystems Considering Many Types of Faults
- Synthesize Fault Prediction, Detection, Isolation, and Response (FPDIR) Plan for the Specified Fault Set
- Conduct Fault and Fault Propagation Modeling, Improve Depth of Time to Criticality Analysis
- Generate Parameter and Sensor Selections
- Conduct False Alarm Analysis
- Refine ECR/FCR Partitioning
- Extensive HMS Design Feature Correlation to Cost and Reliability
- Develop SHM Requirements for Detail Design Phase
- Refine Analysis Models and Simulation/Testbed to Support Cost Effectiveness Evaluation and Design Verification

Table 3.3-11. Preliminary Design Phase Summary

At this point in the design process, the health management system design is well defined. Although there is work ahead in translating the concepts into detailed hardware, the basic framework has been established. For each fault, a plan has been established for its mitigation, parameter selections are being translated into sensor implementation plans, and a layered error and fault containment region plan has been established. Cost effectiveness analyses have been conducted, and economic merits and uncertainties of different fault mitigation ideas have been determined. The simulation of the system has sufficient fidelity to support the major cost effectiveness assessments. Finally, requirements have been further allocated and refined to support the HMS detail design phase.

Table 3.3-12 is a design checklist useful for the preliminary design review. This list, coupled with the list of design product tables for the initial requirements, conceptual design, and preliminary design phases, is a useful checklist to assess the adequacy of the system health management design effort.

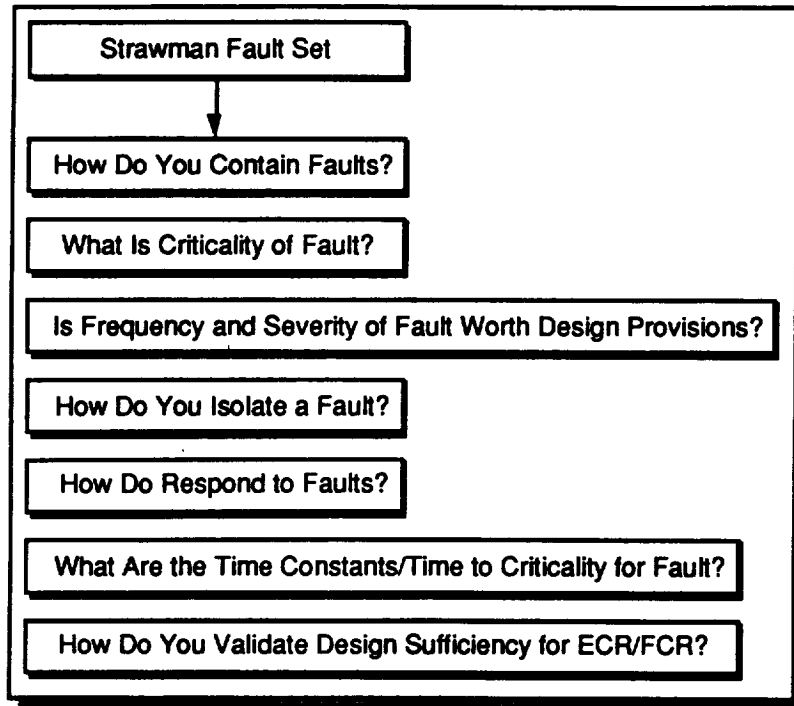


Table 3.3-12. Preliminary Design Review Questions

3.4. SHM Detail Design Phase

3.4.1. Detail Design Phase Objective

3.4.2. Detail Design Phase Major Activities

3.4.3. Detail Design Approach

3.4.3.1. HMS Model Refinement

3.4.3.2. Quantitative Threshold Determination

3.4.3.3. Formal Design

3.4.3.4. Fault Injection Into Detail Design

3.4.3.5. Detail Design Practice for Fault Avoidance

3.4.3.6. Final FMEA

3.4.3.7. HMS/System Integration Detail Design Issues

3.4.3.8. Detailed Data Management Plan

3.4.4. Subsystem Detail Design Issues

3.4.5. HMS Design Support Planning (Training, Personnel, Etc)

3.4.6. Requirements for Fabrication and Test Phase (Test Plan)

3.4.7. Detail Design Phase Summary

3.5. SHM Fabrication and Test Phase

3.5.1. Fabrication & Test Phase Objectives

3.5.2. Fabrication & Test Phase Major Activities

3.5.3. Fabrication & Test Phase Approach

3.5.3.1. Fabrication of HW

3.5.3.2. Verification and Validation

3.5.3.2.1. Analysis

3.5.3.2.2. Testing

3.5.3.2.2.1. Component and Box Level

3.5.3.2.2.2. Subsystem Level

3.5.3.2.2.3. System Level Testing

3.5.3.2.3. Formal Proof

3.5.3.2.4. Simulation

3.5.3.3. Threshold Adjustment

3.5.3.4. Preliminary Operations Planning

3.5.4. Fabrication and Test Phase Summary

3.6. System Deployment and Design Feedback Phase

4.0. Recommendations

4.1. Design Process Tools

4.2. Design Organizational Issues

4.3. Technology Development

4.4. Process Development

The SHM requirements development process should occur in close coordination with the overall system initial requirements development. More work is needed to better develop SHM requirements in a logical and correlated manner. For example, identification and correlation of the primarily quantitative parameters associated with dependable systems such as mean-time-to-repair, fault coverage, mean-time-to-diagnose, etc. are currently very difficult to achieve.

Appendix A. References

- [AIR 4061] "Guidelines for Integration of Engine Monitoring Functions With On-Board Aircraft Systems," AIR 4061 Draft 7, 1992, SAE E32 Engine Monitoring Committee.
- [AND/DORF 91] Musa, J.D., and Ackerman, A.F., "Reliability", Chapt 6.1, Aerospace Software Engineering, Progress In Aeronautics and Astronautics Series, Vol. 136, AIAA, Washington, D.C., 1991.
- [ARP 1587] "Aircraft Gas Turbine Monitoring System Guide Update Draft 4," ARP 1587, March 1992.
- [BOYER 81] Boyer, R.S, and Moore, J.S, "The Correctness Problem in Computer Science," Academic Press, N.Y. 1981.
- [BUTLER 88-1] Butler, R.W., "Estimation of the Distribution of Fault Latency in a Digital Processor," NASA Tech Memo 100521, Nov. 1987.
- [BUTLER 88-2] Butler, R.W. and Sjogren, J.A, "Hardware Proofs using EHDM and the RSRE Verification Methodology," NASA Tech Memo 100669, Dec. 1988.
- [BUTLER 90] Butler, R.W. and Johnson, S.J., "Design for Validation,"
- [Butler, R.W, Di Vito, B.L, and Caldwell, J.L, "Formal Design and Verification of a Reliable Computing Platform for Real-Time Control," NASA.
- [CAO 89] Cao, J. and Wu, Y., "RELIABILITY ANALYSIS OF A TWO-UNIT COLD STANDBY SYSTEM WITH A REPLACEABLE REPAIR FACILITY.", Microelectronics, Reliability, Vol. 29, No. 2, 1989, GB.
- [CARTER 86] -- Carter, W.C. et al, "The System Availability Estimator," FTCS 1986.
- [COCHRAN 91] -- Cochran, K.G., "Artificial Intelligence Techniques Applied to Vehicle Management System Diagnostics," DASC, 1991.
- [COHEN 91] Cohen, G.C, Levitt, K. and Windley, P, "The Formal Verification of Generic Interpreters," NASA Contractor Report 4403, Oct. 1991.
- [CULLYER 88] Cullyer, W.J, "High Integrity Computing," in Formal Techniques in Real-Time and Fault-Tolerant Systems, *Lecture Notes in Computer Science*, Springer-Verlag, NY, 1988.
- [DELISLE 90] Delisle, N, and Garlan, D, "A Formal Specification of an Oscilloscope," IEEE Software, Sept. 1990.
- [DEMARCO 79] DeMarco, Tom., "Structured analysis and system specification," Prentice-Hall, 1979.

- [DEMARCO 84] DeMarco, T, McMenamin, S.M. and Palmer, J.F, "Essential Systems Analysis," Yourdon Press/Prentice-Hall, N.J., 1984.
- [GANE 79] Gane, Chris, "Structured systems analysis: tools and techniques," Prentice-Hall, 1979.
- [GIDEP E885-3027] System Description Methodologies and Formal Specification Languages, RL-TR-91-18.
- [GLASS 79] Glass, R.L, "Software Reliability Guidebook," Prentice-Hall, N.J. 1979.
- [GREUNKE 88] Greunke, R.L, "Guidelines for Worst Case Analysis," Martin Marietta, 1988
- [HALL 90]
- [HARRIS 89] Harris Corp, "WSTA User's Guide", Feb. 1989.
- [HATLEY 87] Hatley, D.J, Pirbhai, I.A, "Strategies for Real-Time System Specification," Dorset House, 1987.
- [HEWLETT 88] Hewlett-Packard, "Structured Methods - An Overview for Engineers and Managers, Hewlett-Packard Corporate Engineering, Palo Alto, CA, 1988.
- [HOLBROOK 90] Holbrook, Capt. H.H. III, "A Scenario-Based Methodology for Conducting Requirements Elicitation," ACM SIGSOFT Software Engineering Notes, vol. 15, no. 1, Jan. 1990.
- [JORD 91] Jordan, J, "NLS Weighting of Evaluation Criteria," NLS System Program Office, 1991.
- [KOPETZ 79] Kopetz, H., "Software Reliability," Springer-Verlag, N.Y. 1979.
- [LAMPORT 80] Lamport, L, Shostak, R, and Pease, M., "The Byzantine Generals Problem," SRI International Computer Science Laboratory, 1980.
- [LLOYD-LIPOW 84] LLOYD, D.K., and Lipow, M., "Software Reliability", Chapt. 17, Reliability: Management, Methods, and Mathematics, 2nd Edition, American Society for Quality Control, Milwaukee, Wi., 1984.
- [LOVELAND 78] Loveland, D.W, "Automated Theorem Proving: A Logical Basis," North-Holland Publishing Co, N.Y., 1978.
- [LUCKHAM 90] Luckham, D, "Programming with Specifications," Springer-Verlag, N.Y., 1990.
- [LUCKHAM 91] Luckham, D, "Two-Dimensional Pinpointing: Debugging with Formal Specifications" IEEE Software, Jan. 1991.

- MIL-STD-1629, Procedures for Performing a Failure Mode Effects and Criticality Analysis
- MIL-STD-1543, Reliability Program Requirements for Space and Missile Systems
- [MUSA 87] Musa, J.D, Iannino, A. and Okumoto, K, "Software Reliability," McGraw-Hill, NY 1987.
- [ORR 77] Orr, K.T, "Structured Systems Development," Yourdon Press, 1977.
- [OSBORNE 92] Osborne, N, "GNC Process Demo," SATWG Meeting Proceeding, 14-16 July.
- [PADULA 91] Padula, S.L. and Sobieszczanski-Sobieski, J, "A Computer Simulaotr for Development of Engineering System Design Methodologies," NASA Technical Support Package for Tech Brief LAR-1350, NASA Langley, 1991.
- [PALMER 92] Palmer, J.D, and Fields, N.A, "An Integrated Environment for Requirements Engineering," IEEE Software, May 1992.
- [RTI 89] "An Evaluation Plan for the Advanced Launch System Multi-Path Redundant Avionics System Architectures," Research Triangle Institute, NASA Contractor Report NAS1-17964, Task No. 28, June 1989.
- [RUSHBY 91-1] Rushby, J., "Formal Specification and Verification of a Fault-Masking and Transient-Recovery Model for Digital Flight-Control Systems," NASA Contractor Report 4384, 1991.
- [RUSHBY 91-2] Rushby, J, von Henke, F. and Owre, S. "An Introduction to Formal Specification and Verification using EHDM," NASA Contractor Report 4384, 1991.
- [SACKS 85] Sacks, I.J, "Digraph Matrix Analysis," IEEE Trans. on Reliability, Vol R-34, No. 5, Dec. 1985.
- SAMSO STD-77-2, FMEA for Satellite, Launch Vehicle and Reentry Systems
- [SHEPPARD 90] Sheppart, J.W. and Simpson, W.R, "Integrated Diagnosis - A Hierarchical Approach," ARINC, 1990.
- [SIEW 91] Siewiorek, D.P., "Architecture of Fault Tolerant Computers: An Historical Perspective," IEEE Paper, IEEE Log Number 9104573.
- [SRIVAS 90] Srivas, M. and Bickford, M, "Formal Verification of a Pipelined Microprocessor," IEEE Software, Sept. 1990.
- [UHRICH 90] Uhrich, D. and Wensley, J, "Reliability and Cost Considerations for Launch Vehicle Avionics," 9th Digital Avionics Systems Conference, IEEE, Oct. 1990.

[VILLEMEUR 91] Villemeur, A., Reliability, Maintainability, and Safety Assessment, Vols 1 and 2, Wiley and Sons, New York, 1992.

[WHITE 83] -- White, A.L., "Reliability with Imperfect Diagnostics," 1983 DASC.

[YOURDON 78] Yourdon, Edward, "Structured design : fundamentals of a discipline of computer program and systems design," Yourdon Press, N.Y., 1978.

Appendix B. List of Acronyms

ALS	Advance Launch System
Anna	Annotated Ada
ARINC	
ATE	Automated Test Equipment
BIT	Built in Test
BITE	Built in Test Equipment
CAE	Computer Aided Engineering
CASE	Computer Aided Systems Engineering
CDS	Command and Data Subsystem
CTV	Cargo Transfer Vehicle
DARPA	Defense Advanced Research Project Agency
DMA	Digraph Matrix Analysis
DOD	Department of Defense
DSE	Dependable Systems Engineering
ECR	Error Containment Region
FA	Fault Avoidance
FCR	Fault Containment Region
FDIR	Fault Detection, Isolation, and Response
FEAT	Failure Environment Analysis Tool
FPDIR	Fault Prediction, Detection, Isolation, and Response
FMEA	Failure Modes and Effects Analysis
FMECA	Failure Modes, Effects, and Criticality Analysis
FOM	Figure of Merit
FT	Fault Tolerance
FTC	Fault Tree Compiler
GIMADS	Generic Integrated Maintenance Diagnostics program
HDBK	Handbook
HM	Health Management
HMS	Health Management System
HW	Hardware
ICD	Interface Control Document
IEEE	
IPT	Integrated Product Team
I/O	Input / Output
K-T	Kepner-Tregoe
LRU	Line Replaceable Unit
LV	Launch Vehicle
MIL-STD	Military Standard
MTA	
MTBF	Mean Time Between Failures
MTP	Maintenance Test Program
MTTR	Mean Time to Repair
NASA	National Aeronautics and Space Administration
NLS	National Launch System

PC	Personal Computer
POINTER	Portable Intelligent Troubleshooter
QFD	Quality Function Deployment
S/C	Spacecraft
SAVE	System Availability Estimator
SHM	System Health Management
STAMP	System Testability and Maintenance Program
STAT	System Testability Analysis Tool
SW	Software
TBD	To Be Determined
TBS	To Be Supplied
TQM	Total Quality Management
TTC	Time to Criticality
VHDL	VHSIC Hardware Description Language
VHM	Vehicle Health Management
V&V	Verification and Validation
WPAFB	Wright-Patterson Air Force Base
WSTA	Weapon System Testability Analyzer

Appendix C. Definitions

Accessability -- A measure of the ease with which an item can be serviced and its work area entered and exited.

Availability -- The availability of a system as a function of time, $A(t)$ is the probability that the system is operational at the instant of time, t .

Avionics -- All the electronic and electromechanical systems and subsystems installed in an flight vehicle or attached to it.

Bug -- A defect introduced into software by a human error in programming.

Common Mode Failures -- Events which cause simultaneous failures in redundant units.

Condition Monitoring -- two particular schemes should be thought of as comprising condition monitoring activities; redline monitoring and health monitoring.

Condition Monitoring provides the data from operation and prestart conditioning necessary to determine if maintenance is required prior to engine start.

Component -- The smallest partition of a system which is considered in system analysis.

Coverage -- Fault detection coverage is the probability that a fault is detected or detectable. Coverage is sometimes used for the conditional probability that given that a fault occurs, the system will recover properly.

Cross-Strapping - An interconnection of functional elements within multistrings to accommodate fault tolerant or information sharing

Criticality -- MIL-STD-1629A: A relative measure of the consequences of a failure mode and its frequency of occurrences.

NASA EG 5320.1: Criticality Category Definitions:

Category	Definition
1	Loss of life or vehicle
2	Loss of mission
3	All others
1r	Redundant hardware element failure of which could cause loss of life or vehicle.
2r	Redundant hardware element failure of which could cause loss of mission.

Criticality Analysis -- A procedure by which each potential failure mode is ranked according to the combined influence of severity and probability of occurrence.

Cut Set -- This term is applied to fault trees to indicate the combination of basic events leading to a failure.

Deductive Approach -- This is the usual approach used in FMEAs (cf. FMEA).

Fault Tree Hdbk: In a deductive system analysis, we postulate that the system itself has failed in a certain way, we attempt to find out what modes of system/component behavior contribute to this failure.

Dependability -- 'Dependable' is a qualitative term that characterizes a system which can be justifiably trusted to deliver the required service whenever needed.

Digraph -- A directed graph or digraph is a collection of a finite number of vertices $P_1..P_n$ together with a finite number of directed edges: P_iP_j in which $i < j$.

Error -- A detectable undesired state. (It exists either at the boundary or at an internal point in the resource, and may be experienced by the user as a failure when it is propagated to and manifested at the boundary)

Fail-Operational -- The ability to sustain a failure and retain full operational capability for mission continuation.

Fail Safe -- The ability to sustain a failure and retain the capability to successfully terminate the mission. For GSE, the ability to sustain a failure without causing loss of vehicle systems or loss of personnel capability.

Failure -- A loss of intended service that is suffered by the user. (designer's or user's intent)

Failure Mode -- A particular scenario in which a failure occurs.

Fault -- The physical or logical cause of an error. An equivalent definition is: "A deviation from desired or expected behavior which may manifest itself as an error." In both definitions, the fault is the prior event which results in some "fault symptom."

Fault Avoidance -- The use of high quality components and conservative design as a means to prevent the occurrence of faults.

Fault Class -- A logical grouping of faults (There is a unifying principle for the grouping).

Fault Containment -- preventing a faulty unit from causing incorrect behavior in a nonfaulty unit.

Fault Coverage -- The ratio of failures detected (by a test program or test procedure) to failure population, expressed as a percentage.

Fault Detection -- Detection of erroneous data within a unit of interest (such as a hardware channel or module). The fault is detected indirectly through the manifestation of an error.

Fault Diagnosis -- The process of isolating a fault and determining its physical cause to the extent necessary to differentiate random faults from generic faults.

Fault Isolation -- The process of determining the location of a fault to the extent necessary to effect repair.

Fault Masking -- A method of accommodating failures that makes the failure transparent to a downstream function.

Fault Set -- The list of all faults which are being considered for a particular purpose.

Fault Tolerance -- Fault tolerance is the survival attribute of a system that allows it to deliver the proper (expected) service after faults have manifested themselves within the system.

FCR -- Fault Containment Regions

A Fault Containment Region is defined as a region of hardware wherein an arbitrary electrical or logic fault does not cause the hardware outside the containment region to misbehave or bail in any manner.

FDIR --Fault Detection, Identification and Response

The process of detecting failures and taking the action necessary to inhibit the failed function and implementing reconfiguration to provide a duplicate non-failed function.

FMEA -- Failure Modes and Effects Analysis -- An analysis of the effects of a loss of an identifiable component.

FMECA -- Failure Mode Effect and Criticality Analysis (FMECA) is essentially similar to a Failure Mode and Effects Analysis in which the criticality of the failure is analyzed in greater detail, and assurances and controls are described for limiting the likelihood of such failures.

Formal Methods - The application of the tools of mathematical logic and formal proof to the verification of computer systems. The various formal methods techniques are based on formal theories, formal specifications and proof.

Formal Specification -- A specification with a mathematical / logical basis. If the specification language is made explicit, then machine aids can be used for analysis.

Generic Fault -- A fault which exists in all copies of the system or redundant components of the system.

Hard Fault -- A permanent change in some component of the system which causes a permanent error.

Hazard -- The presence of a potential risk situation caused by an unsafe act or condition.

Health Management -- Health Management describes the function of assessing and responding to failures within a system. The health management function could consist of on-board detection and responses to faults, maintenance crews, operations teams, or combinations of the above.

Health Management System -- The set of hardware, software, and operations that are implemented for a system to deal with faults. It includes all aspects of the implemented health management, at system, subsystem, and lower levels for a particular system. It is usually implemented as a set of techniques embedded within various subsystems, as opposed to a separate entity.

Health Monitoring -- Health monitoring is defined as the function of detecting and/or predicting failures, and reporting these to the health management system. It is a subset of health management.

Human Error -- The departure of a human operator's behavior from what it should be, this departure exceeding acceptable limits under given conditions.

Inductive Approach -- Inductive methods are applied to determine what system states (usually failed states) are possible; deductive methods are applied to determine how a given system state (usually a failed state) can occur.

Latent Fault -- A fault exists, but does not cause any errors.

Maintainability -- The design attributes that facilitate maintenance.

Maintenance Monitoring -- maintenance monitoring consists of gathering data to determine what repairs, rework or replacement might need to be accomplished to assure the vehicle/system/component will be reliable enough to support its next mission needs. If there is a need for high data rates it should be due to the type sensor used, not due to the criticality of the measurement. An onboard maintenance monitoring system may not necessarily be required for expendable vehicles/engines since a ground system could be used during certain tests (i.e. engine hot fire) to affirm the system is reliable enough to support its mission. Predicting the life remaining in a component, before requiring hardware replacement, is the major goal of maintenance monitoring.

Man Rating -- A man-rated system is one for which all elements are designed with the highest possible reliability, including the required escape system or safe haven. Mission Success and Mission Safety are given equal emphasis.

MTBF -- Mean Time Between Failures -- The MTBF is the mean time between failures in a system with repair, and is thus derived from a combination of repair and failure processes. the easiest approximation for MTBF is $MTBF = MTTF + MTTR$. This expression should be exact for nonredundant systems, but is only approximate for redundant systems because the interplay of multiple failures usually causes the repair rate to change.

MTTF -- Mean Time to Failure -- The MTTF of a system is the expected time of the first system failure in a population of identical systems given successful startup at time zero.

$$MTTF = \int_0^{\infty} R(t) dt$$

Object -- An object is a combination of state and a set of methods which explicitly embodies an abstraction characterized by the behavior of relevant requests. An object is an instance of a class. An object models a real world entity and is implemented as a computational entity that encapsulates state and operations (internally implemented as data and methods) and responds to requests for services.

Operability -- The ability to support required flight rates and schedules by the timely efficient and cost effective solution of all phases of the mission.

PHA -- Preliminary Hazards Analysis is a method for assessing the potential hazards posed, to plant personnel and other humans, by the system.

Preventive Maintenance -- Maintenance carried out at predetermined intervals or according to

Reconfiguration -- Reconfiguration is the dynamic reallocation of redundant elements by executive-level software in response to failure or changes in the aircraft mission.

Redlines -- Redline limits are thresholds used in fault detection in the SSME Controller. These limits may be established above and/or below the nominal operating value for a critical engine parameter. If the limit is consecutively exceeded a specified number of times during the operational phases, then FDIR actions must be taken. Note that redlines may be scheduled, i.e. the limits changed during different phases of the mission.

Redline Monitoring -- Redline monitoring is a term used to identify the process or measuring and testing of parameters in real time (normally at a high rate) to determine whether vehicle/system/component is operating within critical limits. The requirement for high monitoring rates is driven by the fact that the vehicle/system/component can only survive for durations in the order of milliseconds outside of predetermined operating limits. If corrections (reconfiguration, redundancy switching, removal of component or subsystem from operation, etc.) is not accomplished within the millisecond period, catastrophic results will occur. Redline monitoring is a method of determining the instantaneous operating condition of a vehicle/system/component and will be required regardless of whether the vehicle/engine is expendable or reusable.

Redundancy --Protective Redundancy is the set of all elements and functions that make a system fault-tolerant. They could be deleted without reducing system performance in a system that is guaranteed to be free of faults.

Reliability -- Reliability is a measure of the system's ability to provide service even if failures occur within the system. The probability that a system will not fail within time t given that it was operating correctly at time 0.

Repairability -- A measure of the ability to restore an item.

Risk -- to expose to the chance of injury or loss.

Single Point Failure -- Any piece part, assembly, component, or element of construction, such as printed circuit board layout; the failure of which would result in irreversible degradation of item mission performance below contractually specified levels, such as failure of an item in operation that could be catastrophic to a mission objective.

Sneak Circuit Analysis -- An analysis to identify latent paths which cause unwanted functions to occur or which inhibit desired function.

Stand-down -- Post failure stand-down is a period of flight inactivity following a launch or recovery failure.

Symmetric Errors -- when a failure occurs, all parts of the system observe the failure identically.

System -- A system is an assembly of interconnected but separable and independent parts. The system specification is a statement of the social function the system is to perform. System design is a statement of what elements the system will contain and the manner in which they are to be interconnected.

System Health Management -- A term describing the “discipline” of health management for systems in general. It is analogous to a term such as “Propulsion”, which is used to describe the general field of propulsion systems or propulsion engineering. SHM consists of the processes, techniques, and technologies used to design, analyze, build, verify, and operate a system from the viewpoint of preventing or minimizing the effects of failure.

Testability -- The ability to stimulate vehicle hardware or software in order to gather measurement data with which to assess the operability of the vehicle. [Shearer 90-1].

A measure of the ability to determine the functional performance or condition of an item and to fault isolate inoperable or degraded items.

Transient Faults -- A fault which manifests itself temporarily. A transient fault is a one-time event.

Triple Modular Redundancy (TMR) -- An element which is implemented in a triple-modular-redundant (TMR) configuration consists of three identical, independent elements simultaneously performing the same function. The outputs from these three elements are subjected to a majority vote wherever they are used. As a result, the failure of a single element is effectively masked.

Validation -- Validation is the process which attempts to determine if the design meets the abstract operational requirements for the system.

Verification -- Verification is the process which attempts to determine that the implementation of components correctly meets the external characteristics as specified in the top-level design.

Spacer Page Only

Appendix D. Methodology Tools

The concept of a tool exists within a wide scope of application. Computer tools are programs that assist in the design process. Generally referred to as CASE (Computer Aided Systems Engineering or Computer Aided Software Engineering) tools, they are used anywhere from the definition of the design requirements, down to assistance in generating details on hardware and software construction and documentation.

Tools for Initial Requirements Phase

Requirements Development

The accumulation of an initial set of tentative requirements can be accomplished by using a computer program such as Lexscan. This tool analyzes the syntax of natural-language statements. It automatically classifies requirements by applying indexing and clustering techniques. The requirements database can then be analyzed for conflicts, incompleteness, inconsistencies using a Knowledge-based tools. KBRS is a CASE tools designed to perform this function.

The action of producing attributes and identifying requirements often occur concurrently. Attributes are often the result of decomposing pre-existing (but not necessarily accepted) requirements into their components. For example, the requirement for fault avoidance leads to many classes of attributes, ranging from design margins to reliable electronic parts. On the other hand, the requirements associated with each attribute are not always clearly understood at the time the attribute is identified. There may be just a vague knowledge that "this attribute is important" and that something must be done about it.

Holbrook [HOLBROOK-90] suggests a scenario-based methodology of developing requirements. The scenario generation involves producing a design (concept) that approaches the current goals, and describing its behavior. The scenario evaluation phase involves capturing the user's response to the design's behavior. This is a mechanism to uncover unstated requirements, and thus it is important to record this dialog. A hypertext approach is suggested for this process.

Languages for requirements specification have been described. RSL, Requirements Specification Language is based on SREM, the Software Requirements Engineering Methodology. REVS, the Requirements Engineering Validation System includes a translator for RSL.

A tool called OSC for capturing design decisions and supporting information has been described. A motivation is to improve the design review process. Another goal is to develop a reusable design process.[ARANGO 91]

An approach to requirements engineering has been developed at George Mason university. It uses a workstation with hypermedia to provide an integrated environments for requirement development. This environment supports interactive activities between the users and the requirements engineering team which includes requirements elicitation, classification, analysis, traceability, validation and design. [PALMER-92].

Quality Function Deployment

The highest level or most abstract part of the design process is defining what one intends to accomplish, not implementing it. In recent years, focus on system definition has shifted from the process of determining the customers' requirements, to helping the customers understand and define their requirements. Recently, the QFD process has been shown to be an effective methodology. Support tools for this area consist of basic programs, such as Excel. We expect to see more sophisticated tools developed in this area. They should encompass the features of database management, networking, version tracking and object-oriented structure.

The current application of tools to QFD involves customizing of standard spreadsheets and database tools. Limitations of this approach are that modification of the spreadsheets is a continuing process, due to the nature of the problem, and the spreadsheet method is prone to error. New concepts and methods of organizing the information emerge during the process. We see the need for developing specialized tools to assist the process.

Formal Requirements Development

For development of software requirements, a tool referred to as Anna (Annotated Ada) can be used to provide formal specifications [LUCKHAM-91]. Anna extends the Ada language by adding new names and operators. The extensions formally specify the behavior of Ada programs, allowing machine checking of program consistency. The consistency checks could be dumped onto a watchdog processor in order to avoid significant runtime penalties. Work on Anna in this country is concentrated at Stanford. Evaluation models of Anna are available which rely on the use of the Verdex VADS 6.0.3 compiler on a Sun/3 with version 4.0.3 operating system. Stanford plans to offer a one-day tutorial course.

Formal methods research for life-critical flight systems is performed at NASA/Langley. Formal methods use mathematically based analysis to prove that complex systems perform as required. Formal methods consists of formal specifications, implementation and proof. [BUTLER 90]

EHDM attempts to strike a balance between the pure logic of the Boyer-Moore theorem prover and a means of expression that is convenient for humans to use. The language of EHDM is based on first-order predicate logic, but includes some elements of higher-order logic as well [COHEN 91]. It runs on a Sun-3 or Sun-4 Sparc workstation and can be obtained from the Computer Science laboratory of SRI International.

VDM (Vienna Development Method) and Z are European-developed methods. IBM is using Z specification to respecify CICS interfaces to improve maintainability [HALL 90]. Application of Z to an Oscilloscope design at Tektronix has been described [DELISLE 90]. The advantages were perceived to be twofold: a clarification of user requirements, and a basis for the design.

In summary, we are witnessing the broad realization that requirements engineering is an extremely important part of the system design process. CASE tools which address this area are just beginning to appear.

Tools for the Conceptual Design Phase

Since this methodology emphasizes thinking about system failure as early as possible, many tools become relevant at this phase of the design process. The following paragraphs outline some tools that serve as a starting point for the conceptual design analyses and activities.

Design Synthesis

The SHM Conceptual design begins with a set of requirements which were developed and documented in the previous phase. These may or may not presume some degree of partitioning of system functions, i.e. defining subsystem components and/or assigning function(s) to them. However, this partitioning may change during the analysis.

Partitioning of the system requirements is accomplished for several reasons:

- 1) To break the design task into smaller pieces that can be worked on by separate teams so that concurrent design can occur.
- 2) Similar functions are grouped into "subsystems" so that specialists in a small number of technology areas can work together on one subsystem.
- 3) Subsystems are defined with vendor products and capabilities as considerations. The use of existing or similar subsystems is common.
- 4) It may be desired to partition the systems so that different organizations or different companies each receive a certain part of the design task.
- 5) Risk may be lower by having separate organizations design different parts of the system.

The multi-level optimization of the decomposition of a complex design can be aided with a computer simulation tool. The simulator provides a simple function for each subsystem module which models qualitatively the module behavior. Multilevel optimization relies on object or aspect decomposition of a system to break the system optimization tasks into a set of suboptimization tasks and a coordination task which restores the cooperation among the subtasks [PADULA 91].

Data Flow Methodology

The process of partitioning and defining data flow has been addressed by techniques called Structured Methods. Structured Methods are the set of procedures and tools used to create abstract models of the system during its development. Structured Methods consist of Structured Analysis and Structured design, often referred to Yourdon Structured Analysis and DeMarco Structured design. The methodologies have gained wide acceptance and have resulted in the development of CASE tools which facilitate their use.

A structured approach implies a top-down, hierarchical methodology which follows a well-defined procedure. This idea began in the early 1970s with the concept of structured programming. This was, among other things, a thrust to avoid "spaghetti code" in computer programming, mainly through the avoidance of "GOTO"s. Soon after, the approach was applied to design by Warnier and Orr [ORR 77]. The Warnier/Orr methodology was designated DSSD, for data-structured system development.

Structured Analysis looks at the system from a functional standpoint. It is a way of defining a functional specification using a graphical method of *data flow diagrams* (Figure D-1). The purpose of the DeMarco Structured Analysis is to discover the true nature of a problem through the hierarchical decomposition of a system's processes and data [DEMARCO 79].

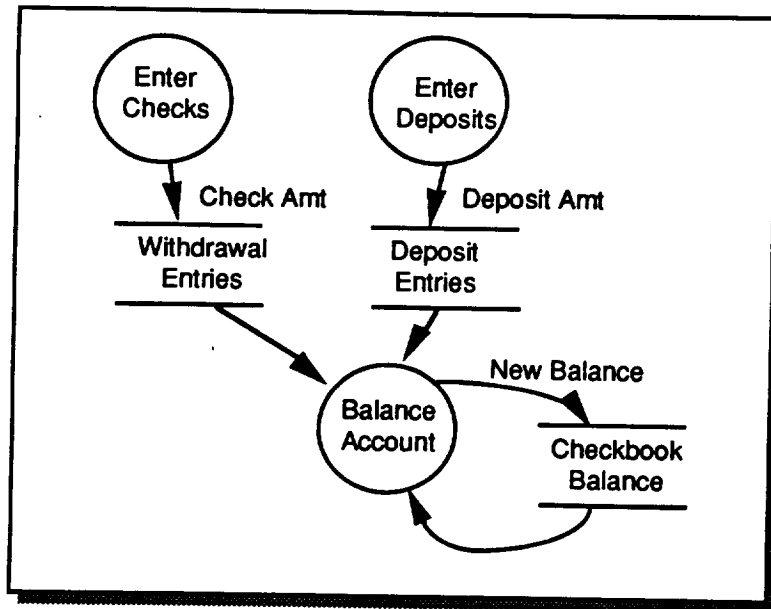


Figure D-1. Example of a Data Flow Diagram

The method described by Pirbhai and Hatley, is described in *Strategies for Real-Time Specification* [HATLEY 87]. This method is based on the DeMarco method of structured analysis[DEMARC0 79]. The methodology uses visual representations of the software at the highest levels of definition. Many CASE tools exist which facilitate this process. We have used *teamwork*™ on the HP and TurboCASE™ on the Macintosh.

Testability Analysis

System Testability is the attribute which:

- 1) Allows the status of a system to be determined
- 2) Provides for isolation of faults

Testability is accomplished in the design phase by incorporating provisions for external and/or built in tests and for monitoring the results of those tests. MIL-STD-2165 describes how testability is incorporated into the various phases of a development program.

Testability tools include:

WSTA	Weapon System Testability Analyzer	HARRIS/NUW C
STAMP	System Testability & Maintenance Program	ARINC
STAT	System Testability Analysis Tool	
POINTER	Portable Intelligent Troubleshooter	ARINC

STAT inputs are items and input tests, test cost, test time. STAT outputs are fault isolation levels, ambiguity group sizes, number of tests required for isolation, test time, test cost, diagnostic test flow diagrams and suggested test types.

STAMP® and POINTER™ are supplied by ARINC to perform Integrated diagnosis. STAMP is used to develop information flow models, assess system testability, develop the diagnostic architecture, and define strategies for BIT, ATE, and manual troubleshooting. POINTER serves as an intelligent controller for BIT, ATE and manual troubleshooting [SHEPPARD 90].

WSTA [HARRIS 89] is a computer program that runs on a Sun workstation. WSTA generates a test strategy which is near optimal in terms of test times or test costs. A primary function of WSTA is to provide static (topological) testability figures of merit, such as average inherent ambiguity group size and feedback loop characteristics. WSTA also provides dynamic test strategy based) figures of merit, such as mean or maximum time to fault isolate. WSTA provides guidance to the designer on the optimal placement of test points based on the fault isolation data each test point can provide. WSTA utilizes a system dependency model and the time-efficient sequencer of tests (TEST) algorithm to generate an optimal test strategy. WSTA uses a readily extendible repertoire of user-selectable diagnostic strategies. Incorporates a fault simulation and test sequencing process based upon information theory to create an optimum initial test tree.

FMEA/FMECA Tools

FAULT TREE ANALYSIS — The FEAT tool can be used to build fault trees. The digraph form is used to represent the fault tree. The analysis of singleton and doubleton faults is automated. FEAT models are developed on a Macintosh with MacDraw. Digraph and Schematics are linked so that failed nodes can be seen directly on the schematic. Schematics can be electrical, mechanical, hydraulic, etc.

The Fault Tree Compiler is supplied through NASA Langley. It runs on a Sun workstation. FTC was developed as a faster method of solving fault tree probabilities to replace the use of CARE III.

MIL-STD-1629 FMEA/FMECA are automated with the program PC - FMEA/FMECA which runs on an IBM PC and is produced by Management Sciences, Inc. of Albuquerque NM.

Cost vs Reliability Modeling

Cost models which include reliability considerations are used in order to predict life cycle costs. The models allow comparison of different possible levels of redundancy against cost. Redundant elements can be used to mask faults, or to provide reconfiguration which eliminates faults from the functional part of the system. The cost of providing this fault protection is extra hardware, and complexity. The advantages of redundancy must be evaluated against cost and probability of failure. These trade studies are performed early in the design process and refined as the design becomes more definitized [UHRICH 90].

Reliability Modeling

Failure Probabilistic Analysis — The probability of failure for critical items can be automated using Digraph Matrix Analysis (DMA), a program by RDI Associates. [SACKS 85]. Perhaps future upgrades of FEAT will also allow this. FEAT can be integrated with CLIPS which then can provide automated calculation of failure paths.

Reliability Prediction — MIL-STD-756 prediction is performed on the 756 PREDICT program from SYSCON which runs on an IBM/PC or on VAX/VMS. This tool uses a deterministic combinatorial approach instead of Monte Carlo or Markov simulation.

Reliability Analysis — MIL-HDBK-217 prediction of failure rates can be calculated using ARM (Advanced Reliability Modeling) tool, from Confidence Enhancements Ltd., Batavia IL. This runs on IBM/PC or VAX/VMS UNIX.

Dormant Failure Rates — Trade studies should be made to compare failure rates for continuous on versus keeping part of the system OFF. This could be applied to backup components, with periodic turn-on and testing. The program DORMACALC from Sendrian Resources, Newbury Park CA, calculates non-operating failure rates.

Failure Rate Prediction for Reconfigurable Systems — SURE/ASSIST from NASA Langley runs on the Sun workstations. It computes Markov models for systems which reconfigure in response to fault detection. Inputs are failure rates and reconfiguration times.

SYSTEM AVAILABILITY — Monte Carlo analysis of system availability can be performed by SAVE (System Availability Estimator). [CARTER-86]

BELLCORE performs MIL-HDBK-217 calculations and includes infant mortality. It can incorporate laboratory data and field data. This is from Bell Communications Research.

MILSTRESS performs MIL-HDBK-217 calculations and allows input of new component data not included in MIL-HDBK-217. Runs on IBM/PC, VAX/VMS. POC is Mitchell & Gauthier Assoc., Concord, MA.

ORACLE — Optimized Reliability and Component Life Estimator from ROME Laboratory/RBET runs on an IBM/PC or VAX/VMS. It automates MIL-HDBK-217 and allows series and parallel components.

RECALC2 — Automates MIL-HDBK-217. Inputs include a system hierarchy, redundancy and circuit data. Runs on an IBM PC. From T-Cubed Systems, Westlake, CA.

Performance Modeling Tools

Performance Modeling tools include the range from low-level circuit analysis using SPICE to high-level system behavior models. At the highest level, these tools provide behavior modeling of the system.

At the highest levels of abstraction we can use performance modeling to simulate independent interactive events in order to predict the probability of entering various system states. Discrete event simulation is used to run repeated scenarios on a system with various random inputs. The results are analyzed statistically to determine system characteristics. These results can be used to determine whether the system elements have adequate performance, and whether or not sufficient redundancy is used. Typical parameters determined are "timing and sizing" estimates. For example, the effect of bus traffic on selection of the number of buses and their required throughput rate can be determined in this manner. Tools used for this type of simulation are GPSS, ADAS, SES/workbench, Teamwork/SIM, ...

ADAS is a tool developed by RTI for the hierarchical description and assessment of system designs. In ADAS, the system performance model is created from a structural model of the architecture and a data/control flow model of the processes [RTI 89].

Lower-level simulation involves detailed performance of circuit elements. In the digital area an integrated tool such as VHDL might provide the hierarchical description of the system which allows modeling at various levels. For analog or mixed signal, a more detailed modeling is provided by tools such as PSPICE, SABER, ...

Modeling at the functional level is performed by tools such as MATLAB/SIMULAB for analog simulation and SES/workbench for discrete event simulation. SIMULAB allows you to define the system using block diagrams. The blocks can contain transfer functions, signal or noise sources, transport delays, integrators, zero-order holds, and varying sample rates, etc. Block diagrams are constructed from objects and interconnected on a screen.

Design Verification and Validation

Case Analysis — Martin Marietta has a documented methodology for Worst-Case Analysis [GREUNKE 88]. MMC uses PSPICE, SABER, SUPER-COMPACT, HILO-3 and other CAE tools. The worst case analysis activities address both digital and analog designs. Testing may be used to supplement the use of CAE tools.

Formal Verification

Formal methods can be applied in hardware. A case in point is the 32 bit VIPER microprocessor which has been formally verified [BUTLER 88-2], [CULLYER 87]. The VIPER was designed at the Royal Signals and Radar Establishment (RSRE), Malvern, UK. One of these was delivered to NASA Langley for their work in formal verification. The VIPER modules were specified in the hardware description language ELLA and in HOL. The HOL description was verified at Cambridge using a theorem prover on a Sun workstation.

Odyssey Research Associates used formal methods to specify and verify a microprocessor, the Mini Cayuga [SRIVAS 90]. The verification was performed with Clio, a functional-language based verification system.

Many investigators were motivated towards formal methods by problems which seem to elude simple logical analysis. Such is the Byzantine General's problem [LAMPORT 80], which arises when multiple processors are used for fault tolerance, and the decision process must be distributed among the processors. Several studies of this problem have used formal methods [RUSHBY 91-1].

Temporal logics have been proposed for verifying concurrent operations. Machine checking of temporal logic specifications of independent processes, such as we have with the FDIR of reconfigurable systems, provides an alternative to using modeling for verification.

Design Synthesis tools for performing analysis of detailed circuits, logic, etc. have been available. The current thrust to provide an Integrated Engineering Environment.

Design Database for HMS

A key to implementation of this Methodology lies in retaining the knowledge gained/decisions made throughout each engineering design phase. A good HMS design requires simplification of the monitoring of product development from a multi-disciplinary perspective. This multi-functional team coordination with a common visibility to all activities and data can best be accomplished by creating a way to capture all engineering decisions, analyses and data in a shareable electronic media. Such a concept has been demonstrated by the Advanced Technologies GN&C IRAD Group at Martin Marietta Astronautics Group [OSBORNE 92].

Tools for other phases---TBD

Appendix E. Methodology Summary

This Appendix summarizes in Tables (E-1, E-2, E-3, E-4, E-5, E-6) the major products and activities for each of the SHM design phases discussed in this document: 3.1 Initial requirements, 3.2 Conceptual design, 3.3 Preliminary design, 3.4 Detail design, 3.5 Fabrication & Test, and 3.6 Deployment & Operations. Sections 3.4, 3.5 and 3.6 in this revision of the document are incomplete, however, top level lists of the major design products of these phases have been included in this appendix.

Customer/Contractor Generated Weighting Factors and Customer Demands.
SHM Related Constraints and Figures of Merit
Fault Set Classes for Hardware, Software, and the Operations Elements
SHM Requirements At Appropriate Level to Conduct Subsystem Level Trade Studies
Contribute SHM Insight Into Launch System Design Functional Analysis Concept

Table E-1. Products and Activities of the Initial Requirements Phase

- Fault Set Definition Refinement for Application At Subsystem Level
- Lists of Faults (Within Subsystems At Best Fidelity Known) To:
 - Design Out (Identification of Basic Approach How)
 - Inspect Out
 - Tolerate (Identify Conceptual Active or Passive FT Approach)
 - Monitor/Report Only
- Parameters To Monitor:
 - Desired Parameters for Predictive Trending
 - For Active Fault Tolerance
 - Parameters for Life Usage Calculations
 - Ground Checkout Parameters
- Major Timing Constraint/Requirements Identification (Time to Criticality)
- Error/Fault Containment Region (ECR/FCR) Partitioning Between Subsystems and At Major Levels Within Subsystems
- Health Management Data Flow Plan
- Formulation of Overall Verification and Validation Plan. Rough Identification of Special Provisions for Verification In SHM System Design.
- Degree of System Autonomy
 - Ground Based Vs. Flight Based Decision Partitioning
 - Degree of Human Role In SHM Functions
- SHM Conceptual Design Software Requirements Need to Be Input to Overall Vehicle Software Plan/Requirements
- Most Passive Fault Tolerant Concepts Must Be Identified Early and Appropriate Up Front Design Provisions Made
 - Margins, Special Materials for Delayed Failure, Fail On Selected Paths
 - Fire Walls, Self Sealing Tanks, Physical Containment & Separation
- Develop Requirements for Simulation/Testbed Design
- Refine Analysis Tools for Design Support
- Develop SHM Requirements for Next Phase

Table E-2. Conceptual Design Products

- Complete Fault Prediction, Detection, Isolation, and Response (FPDIR) Plans Hardware, Software, and Algorithmic Approach
 - Levels of Redundancy
 - Degree of Cross Strapping
 - Utilization of Voting, Hot or Cold Standby, Particulars of Redundancy Implementation
 - Threshold and Persistence Levels for Alarms
 - Refinement of Parameter List, Combining Parameters With Data Fusion for Information Confidence
- FPDIR Design Analyses:
 - Detailed FMEA and Augmentation of FMEA With:
 - Quantitative Failure Rate Estimates
 - Gathered Fault Combination Analysis
 - Fault Propagation Analyses
 - False Alarm Analyses
 - Time to Criticality (Using Fault Propagation Analyses)
 - Cost Effectiveness and Reliability Analyses
- Preliminary Sensor Selection for Parameters
- Design Specifics for Layered Error/Fault Containment Regions (ECR/FCR)
 - Hierarchy of Service and Control
 - Partitioning and Allocation of Functions and State Variable Sets
 - Communications Protocol
 - Hardware and Software Partitioning, Fault Classes for Partitioning
 - Isolation Requirements and Mechanisms
- Refinement of Architecture Concepts at System and Subsystem Level
 - Network Nodal Arrangement, Protocol, and Timing
 - Modes of Operation of Each Subsystem and Mode Compatibility
 - Signal Conditioning, Multiplexing, Data Processing, Data Storage/Retrieval
 - Identification of Requirements for Control System Performance In Off-Nominal Fault Induced State
- Provisions for Retest/Recovery/Reintegration for Switched Out Components and Subsystems
- For Fault Responses of Design Out, Inspect and Checkout; and Detect and Report Only, Identification of Personnel, Equipment, Training, Procedures, and Other Means To Implement These Procedures
 - Quantified Requirements for Inspection and Test
 - Human Interface Requirements
- Test Bed/Simulation Setup
- Refinement of Health Management Data Plan
- Special Provisions for Ground System Health Management
- Requirements Refinement for Next Design Phase

Table E-3. SHM Preliminary Design Activities and Products

- **Refine Models and Conduct Detailed Cost Effectiveness Analyses On HMS Design**
- **Develop Detailed Algorithms for FPDIR, Develop & Test Quantitative Thresholds With Help of Simulation/ Testbed**
- **Inject Faults Into Simulation/Testbed, Characterize/ Verify HMS Design Effectiveness**
- **Fully Integrate the HMS Design with the Full Launch System Design**
- **Generate HMS Detail Design for All Elements**

Table E-4. Products of Detailed Design Phase

- **Design Produced**
 - Drawings and Documentation
 - Parts Manufactured and Assembled
 - Software Code Refined
- **Refined FMEA**
 - Interaction Faults Better Understood Via Test
- **Utilization of Specific Failures for Quantitative Verification (Design Meets Requirements) and Validation (Design Intent Met)**
- **Adjust Thresholds to Control Response Sensitivity and False Alarm Rate**

Table E-5. Products of Fabrication and Test Phase

- **Train Personnel on HMS System Utilization**
- **Refine Design to Accommodate Field Experience Lessons Learned**
- **Refine HMS Design If Subsystem Design Changes Are Made**
- **Adjust Thresholds to Control Alarm Response Sensitivity and False Alarm Rate**

Table E-6. Products of Deployment and Operations Phase

A top level design review question set is presented in Table E-7. The design review questions should be useful to those responsible for the SHM design to assure that the HMS design activity has been given appropriate system design attention.

- **What Are the Fault Classes Addressed By the Design?**
- **How Are the Following Fault Classes Treated?**
 - Transient Faults - Latent Faults
 - Intermittent Faults - Permanent Faults
- **Has the FMEA/CIL Been Augmented By Quantitative Estimates of Fault Frequency?**
- **Has the Fault Accomodation Plan Been Documented for Each Fault, Including Design Out, Inspect/Test Out, Fault Tolerance, & Detect Record Only?**
- **For Faults Requiring An Active Response Mechanism, How Do You Detect, Isolate, and Respond?**
- **Has A Layered ECR/FCR Implementation Considering the Time Domain and Boundary Domain Been Established?**
- **How Is Data Integrity Protected From Damage Caused By Faults?**
- **How Do You Validate Design Sufficiency for ECR/FCR?**
- **What Is the Response Time to Faults Compared to Associated Time to Criticality?**

Table E-7. SHM Design Review Questions

—

.

—

—